Universidade de Aveiro Departamento de Matemática
2005

Nuno Lopes Martins **Classificação e partição de polígonos simples**

Universidade de Aveiro Departamento de Matemática
2005

Nuno Lopes Martins

Classificação e partição de polígonos simples

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática, área de especialização em Ensino, realizada sob a orientação científica do Professor Doutor António Leslie Bajuelos Domínguez, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro.

Ao meu Pai que está e estará sempre presente.

o júri

presidente

Professora Doutora Maria Rosália Dinis Rodrigues
Professora Associada da Universidade de Aveiro

Professora Doutora Ana Maria Carvalho de Almeida
Professora Auxiliar da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Professor Doutor António Leslie Bajuelos Domínguez
Professor Auxiliar da Universidade de Aveiro

agradecimentos

Ao Professor Doutor António Leslie Bajuelos Domínguez pela preciosa orientação prestada na realização deste trabalho. Um muito obrigado pelas sugestões, esclarecimentos e pela total disponibilidade que sempre manifestou ao longo do seu desenvolvimento.

Aos meus amigos, Humberto e Rui, pelas ajudas úteis que me prestaram.

A todos aqueles que, de um forma directa ou indirecta, contribuíram para que esta dissertação se tornasse possível.

palavras-chave

Polígonos simples, partição de polígonos, triangulação, quadrangulação, pseudo-triangulação, involúcros convexos, polígonos ortogonais.

resumo

Esta dissertação tem como objectivo fazer um estudo sobre polígonos simples, nomeadamente no que concerne à sua classificação e partição. Começa-se por apresentar várias classes de polígonos simples fazendo depois uma classificação hierárquica. São apresentados alguns exemplos de polígonos simples segundo algumas características específicas. Posteriormente aborda-se o tema da partição clássica de polígonos simples. Faz-se uma resenha histórica sobre a evolução da complexidade da triangulação de polígonos simples, apresentam-se os algoritmos mais marcantes deste tipo de partição e mostra-se como, a partir de polígonos simples triangulados, se pode obter uma quadrangulação. Faz-se, também, uma abordagem a uma partição não clássica, como é o caso da pseudo-triangulação. Por fim, apresentam-se alguns problemas que ainda permanecem em aberto.

keywords

Simple polygons, polygons partition, triangulation, quadrangulation, pseudo-triangulation, convex-hulls, orthogonal polygons.

abstract

The goal of this dissertation is to study simple polygons, namely concerning their classification and partition. We start by presenting several classes of simple polygons, performing next a sorted classification. Some examples of simple polygons are presented according to some specific characteristics. The classical partition of simple polygons theme is discussed next. We make an historical draft on the evolution of the triangulation complexity of simple polygons, the fundamental algorithms of this type of partition are described, and its shown how, starting with simple triangulated polygons, we can obtain a quadrangulation. An approach to non-classic partitions is done, e.g. the pseudo-triangulation. At last, some problems that remain unsolved are presented.

Conteúdo

Lista de Figuras

Lista de Notações

$a \equiv b$ — resto da divisão inteira de a por b.
$\lceil x \rceil$ — menor inteiro maior ou igual a x.
$\lfloor x \rfloor$ — maior inteiro menor ou igual a x.
$|E|$ — número de elementos do conjunto E.
$[uv]$ — segmento de recta uv.
$A \setminus B$ — elementos de A que não pertencem a B.

Capítulo 1

Introdução

A Geometria Computacional é uma área das Ciências da Computação que surgiu em meados dos anos 70. Apesar de ser uma área recente, tem raízes bem antigas, baseadas na Geometria Euclidiana. No entanto, só em 1985 foi publicado o primeiro livro sobre o assunto, *"Computational Geometry: An Introdution"* escrito por *F. P. Preparata* e *M. I. Shamos* [85].

Existe uma grande semelhança entre a Geometria Computacional e o Desenho Geométrico, se tivermos em conta que ambos pretendem obter novos elementos geométricos a partir de construções elementares. A diferença está no facto de que, as figuras geométricas e construções correspondem a estruturas de dados e algoritmos. Em geral, o interesse está em solucionar um problema utilizando o menor número possível de operações elementares de modo a promover soluções algorítmicas diferentes. Assim sendo, o principal objectivo da Geometria Computacional é estudar problemas geométricos sob o ponto de vista algorítmico.

Tem-se desenvolvido como uma área importante que conta com cada vez mais investigadores. Esta adesão pode ser, em parte, explicada pelos atractivos problemas com enunciados simples de compreender. Apesar desta aparente simplicidade, a Geometria Computacional, constitui uma ferramenta fundamental em diversas áreas de computação que recorrem a abordagens geométricas, como por exemplo:

- Computação Gráfica: é a parte da computação destinada ao uso de imagens em geral. Vários problemas de Geometria Computacional são motivados por

problemas geométricos que aparecem em Computação Gráfica. Por exemplo: 1) Ao seleccionarmos um objecto num interface gráfico, devemos seleccionar de entre todos os objectos desenhados na tela, aquele que está mais próximo de uma determinada posição. 2) Para desenhar, de forma realista, uma cena tridimensional, é necessário saber como os vários objectos se projectam na tela e tratar as suas oclusões. 3) Para fazer uma animação realista, é necessário detectar se há colisões entre objectos que se movem e o resto da cena.

- Robótica: ramo da Mecânica que actualmente trata de sistemas compostos por máquinas e partes mecânicas automáticas, controlados por circuitos integrados (micro processadores), tornando sistemas mecânicos motorizados, controlados manualmente ou automaticamente por circuitos ou mesmo por computadores. Um dos problemas fundamentais em robótica é o planeamento de movimentos. O robot precisa de analisar o seu ambiente e descobrir uma forma de se mover de um ponto ao outro sem colidir com os objectos do ambiente. Além disso, ele quer fazer isso da forma mais eficiente possível, o que implica na necessidade de identificar o caminho mais curto viável entre os dois pontos.

- Sistemas de Informação Geográficas (SIG´s): é um sistema de hardware, software, informação espacial e procedimentos computacionais, que permite e facilita a análise, gestão ou representação do espaço e dos fenómenos que nele ocorrem. Estes sistemas lidam com enormes quantidades de dados geométricos para poder representar fielmente a geometria de estradas, rios, fronteiras, curvas de nível, áreas de vegetação, etc. Um problema típico nesta área é saber que objectos geográficos estão perto uns dos outros. Por exemplo, se um rio ameaça transbordar, que cidades e estradas serão afectadas?

- Desenhos de Circuitos Integrados (VLSI[1] "artwork"): estes circuitos são compostos por dezenas de milhares de componentes electrónicos que não se podem sobrepor, para evitar curto-circuitos. Durante o projecto desses circuitos é necessário identificar, de uma forma eficiente, sobreposições que possam existir.

Entre outros exemplos, como sejam: Optimização Combinatória, Processamento de Imagens, Teoria de Grafos, Desenho Auxiliado por Computador (CAD[2]), etc.

[1] Very Large Scale Integration.
[2] Computer Aided Design.

De entre os vários problemas que são objecto de estudo em Geometria Computacional, existe um que é o da partição de polígonos que merecerá particular atenção nesta dissertação. Em Geometria Computacional, algoritmos para problemas de partição de um polígono arbitrário, são, de um modo geral, mais complexos que aqueles algoritmos para polígonos com determinadas características, tais como, por exemplo, os convexos ou os estrelados. Uma estratégia para resolver alguns destes problemas quando é usado um polígono arbitrário, é particioná-lo em componentes mais simples, resolver o problema para cada componente usando um algoritmo específico e depois combinar as soluções parciais.

Esta dissertação divide-se em cinco capítulos. Após uma introdução feita neste capítulo, começamos, no capítulo 2, por fazer a definição de vários tipos de polígonos simples apresentando, depois, uma classificação desses polígonos. Nos capítulos seguintes fazemos a divisão da partição de polígonos, em clássicas (capítulo 3) e não clássicas (capítulo 4). Nas partições clássicas são tratados os temas da triangulação e da quadrangulação de polígonos simples. O problema da triangulação consiste em, dado um polígono simples qualquer, P, particionar o polígono em triângulos com interiores disjuntos, de tal forma que as arestas desses triângulos são tanto arestas como diagonais de P, unindo pares de vértices de P. Apesar de se usar na maioria das situações triângulos na partição de polígonos, existem, no entanto, outras formas de o particionar, usando, por exemplo, quadriláteros (quadrangulação). Nas partições não clássicas temos a pseudo-triangulação. Abordaremos este e outros conceitos, como o de pseudo-triangulações mínimas e pseudo-triangulações minimais. Apresentaremos alguns resultados importantes e algumas relações entre tamanhos de pseudo-triangulações. No capítulo final apresentaremos problemas que ainda se encontram em aberto. De referir ainda que ao longo da dissertação somente se abordam polígonos simples, pelo que sempre que o termo "simples"for omitido, o termo polígono refere-se a polígono simples.

Capítulo 2

Polígonos simples: classes e classificação

2.1 Algumas classes de polígonos simples

Nesta primeira secção, após definirmos o conceito de polígono simples, apresentamos a definição de vários tipos de polígonos simples e mostramos alguns resultados mais relevantes relacionados com alguns desses polígonos.

Definição 2.1.1 *Define-se* ***polígono simples****, P, para qualquer inteiro $n \geq 3$, no plano euclidiano $\mathbb{R}^2$, como sendo a figura $P = [v_1, v_2, ..., v_n]$ formada por n pontos $v_1, v_2, ..., v_n$ em $\mathbb{R}^2$ e por n segmentos de recta $[v_i, v_{i+1}]$, $i = 1, 2, 3, ..., n-1$ e $[v_n, v_1]$. Aos pontos v_i chamamos* ***vértices*** *do polígono e aos segmentos de recta,* ***arestas****. Um polígono simples fica bem definido se e somente se:*

1. *a intersecção de cada par de segmentos adjacentes é um e um só vértice, isto é, $[v_i, v_{i+1}] \cap [v_{i+1}, v_{i+2}] = v_{i+1}$; (note-se que os índices são considerados módulo n; por exemplo $v_{n+1} \equiv v_1$).*
2. *Segmentos que não sejam adjacentes não se intersectarem.*

Por definição assumimos que as arestas $[v_{n-1}, v_n]$ e $[v_n, v_1]$ são adjacentes.

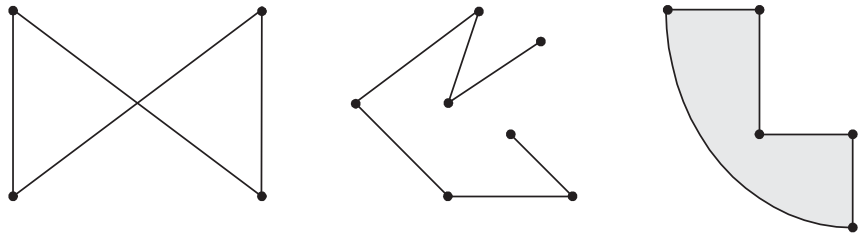


Figura 2.1: Exemplos de figuras que não são polígonos simples.

A definição de polígono simples não é única. Podemos, por exemplo, defini-lo recorrendo ao conceito de cadeia poligonal.

Definição 2.1.2 *Chamamos* ***cadeia poligonal*** *a um conjunto de n pontos distintos do plano* $v_1, v_2, ..., v_n$*, chamados vértices, ligados por segmentos,* $[v_1, v_2], [v_2, v_3], ..., [v_{n-1}, v_n]$*, as arestas.*

Se arestas não adjacentes não se intersectarem, então dizemos que a ***cadeia poligonal*** *é* ***simples****.*

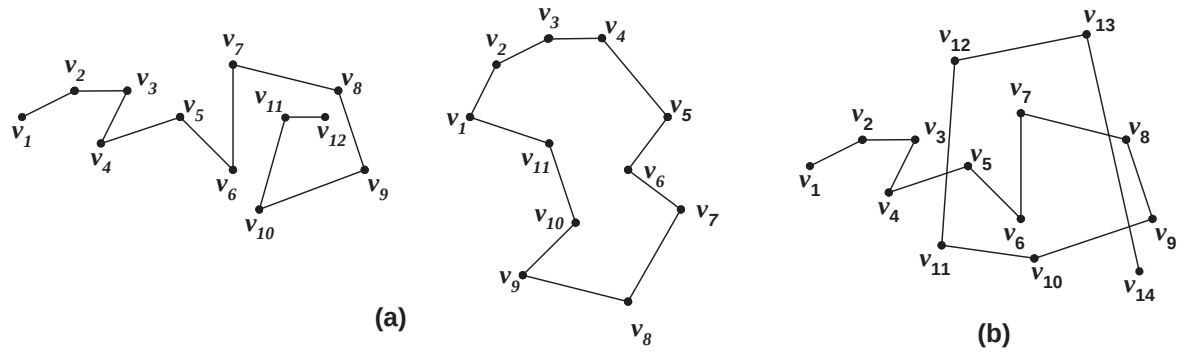


Figura 2.2: (a) Cadeias poligonais simples; (b) Cadeia poligonal não simples.

Sempre que na cadeia poligonal simples v_1 e v_n estiverem ligados por uma aresta, estamos na presença de uma **cadeia poligonal fechada**. Esta cadeia poligonal determina uma curva de *Jordan* (curva fechada sem auto-intersecções) que nos divide o plano em duas regiões: uma interior à curva, outra exterior[1].

[1]Teorema da Curva de Jordan: Toda a curva simples e fechada divide o plano em duas regiões.

Podemos, assim, dar uma definição diferente da dada na definição 2.1.1.

Definição 2.1.3 *Chamamos* ***polígono simples*** *ao conjunto dos pontos da região interior reunidos com os pontos da cadeia poligonal simples fechada.*

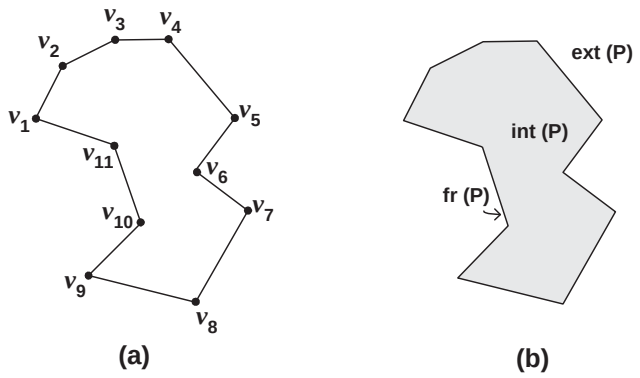


Figura 2.3: (a) Cadeia poligonal simples fechada;
(b) Divisão do plano que foi originada pela cadeia.

Definição 2.1.4 *Chamamos:*

1. *Ao conjunto de todos os pontos interiores de P,* ***interior de P****, $int(P)$.*
2. *Ao conjunto de todos os pontos pertencentes aos segmentos de recta,* ***fronteira de P****, $fr(P)$ ou ∂P.*
3. *Ao conjunto de todos os pontos exteriores a P,* ***exterior de P****, $ext(P)$.*

Um polígono simples, P, fica perfeitamente determinado por o conjunto formado pelos seus vértices ordenados, quando se percorre a fronteira de P, e por uma orientação que nos permita conhecer onde se irá situar o interior de P. Assim, se percorrermos a fronteira no sentido negativo (horário), encontramos o interior de P à direita de qualquer aresta, caso contrário, no sentido positivo (anti-horário), o interior de P, encontrar-se-á à esquerda de qualquer aresta.

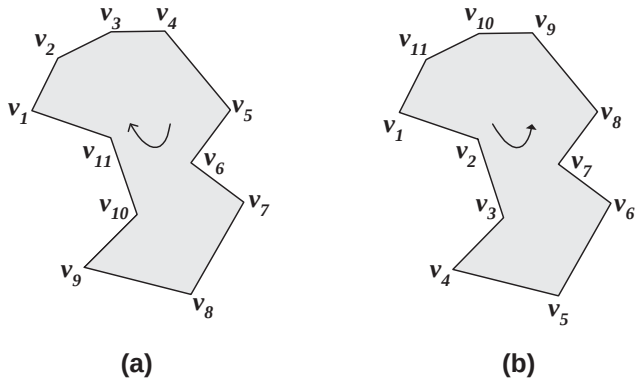


Figura 2.4: (a) Polígono orientado no sentido negativo;
(b) Polígono orientado no sentido positivo.

Por norma, suporemos sempre que os vértices de um polígono simples estão orientados no sentido positivo (anti-horário).

Um **vértice** de um polígono diz-se **convexo** se a amplitude do ângulo, pertencente ao seu interior, formado por duas arestas que lhe são incidentes for menor ou igual a π. Caso contrário o vértice diz-se **côncavo** ou **reflexo**.

Lema 2.1.1 (Meister [74]) *Qualquer polígono simples, P, tem um vértice estritamente convexo.*

Prova: Seja P um polígono. Se percorrermos a fronteira de P, no sentido anti-horário, quando encontramos um vértice estritamente convexo, temos que virar à esquerda e num vértice estritamente côncavo viramos à direita. Seja v o vértice de P com ordenada mínima e abcissa máxima. Seja l a recta horizontal passando sobre v. A aresta seguinte a v, que lhe está incidente, está acima de l (ver figura 2.5). Logo, deveremos virar à esquerda em v. Portanto, v é um vértice estritamente convexo. ■

Outro conceito importante, também pela sua aplicação, (computação gráfica [67], robótica [62, 13], planeamento de movimentos [82, 63], reconhecimento de padrões [109]) é o de **visibilidade**.

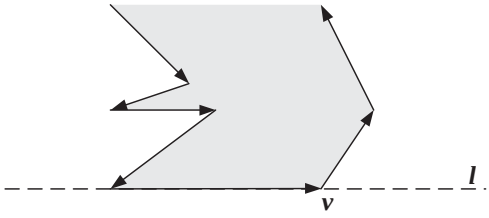


Figura 2.5: Ilustração da prova do lema 2.1.1.

Definição 2.1.5 *Dizemos que* ***dois pontos*** x *e* y*, num polígono simples,* P*, são* ***visíveis*** *se nenhum ponto do segmento de recta* xy *pertencer ao* $ext(P)$*.*

Podemos dizer, usando uma terminologia diferente, que x vê y se x e y são visíveis.

A definição 2.1.5 permite que o segmento de recta xy intersecte um vértice côncavo ou percorra mesmo uma aresta (ver figura 2.6).

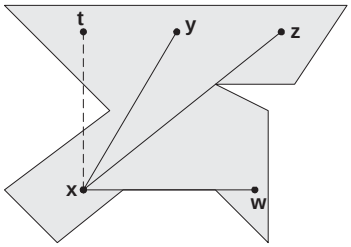


Figura 2.6: O ponto x vê os pontos y, z e w mas não o ponto t.

Definição 2.1.6 *Um polígono simples,* P*, diz-se* ***convexo*** *se para todo* x*,* $y \in P$*, o segmento* $[xy] \subset P$*.*

Podemos também definir polígono simples convexo recorrendo ao conceito de visibilidade:

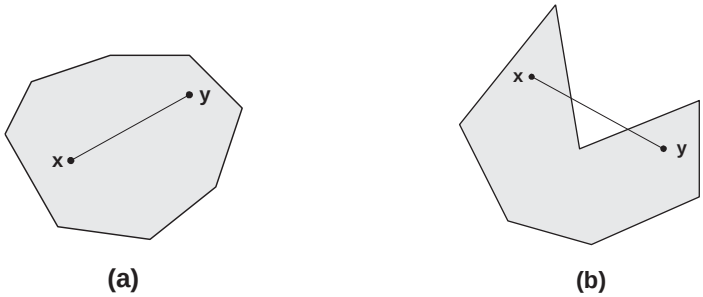


Figura 2.7: (a) Polígono convexo; (b) Polígono não convexo.

Definição 2.1.7 *Um polígono simples é convexo se qualquer par de pontos pertencentes ao seu interior ou à fronteira, são visíveis, isto é,*

$$\forall\, x, y \in int(P) \cup fr(P),\ [xy] \cap ext(P) = \emptyset.$$

Teorema 2.1.1 *Um polígono é convexo se e só se não tem vértices côncavos.*

Prova:

($\Leftarrow$)

Se o polígono não tem vértices côncavos, então dados dois quaisquer pontos de P, o segmento que os une pertence a P, logo P é convexo pela própria definição de polígono convexo.

($\Rightarrow$)

Suponhamos, por absurdo, que P tem vértices côncavos, então existem pelo menos dois pontos, tal que o segmento que os une, não pertence na totalidade a P, como por hipótese P é convexo, quaisquer dois pontos de P, podem ser unidos por um segmento que pertence a P, temos assim um absurdo. Logo P não tem vértices côncavos. ■

Definição 2.1.8 *Seja S um subconjunto de pontos de $\mathbb{R}^2$. Chamamos* ***invólucro*** *ou* ***fecho convexo*** *ao menor conjunto convexo do plano que contém S.*

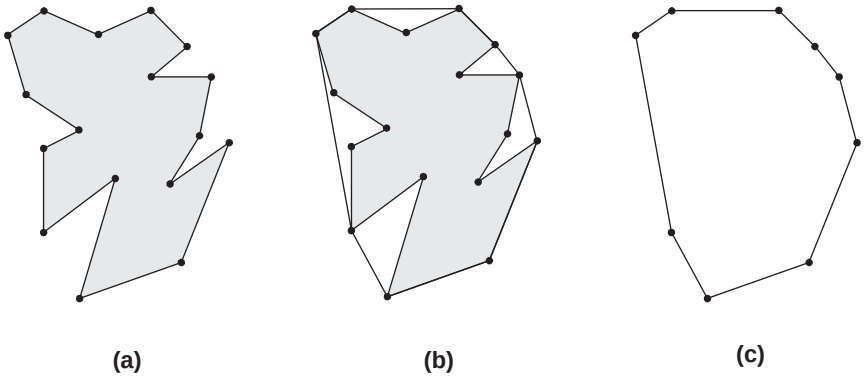


Figura 2.8: (a) Polígono simples inicial; (b) O invólucro convexo do polígono; (c) Polígono convexo.

Definição 2.1.9 *Chamamos* ***invólucro*** *ou* ***fecho convexo de um polígono*** P, $CH(P)$ *(do inglês, convex hull), ao menor polígono convexo que contém* P.

Definição 2.1.10 *Chamamos* ***bolso*** *de um polígono simples,* P, *a uma região limitada, exterior a* P *mas interior do invólucro convexo de* P.

O bolso é limitado pelo polígono. As arestas deste novo polígono formado, são também arestas de P, excepto uma aresta exclusiva do invólucro convexo. A esta aresta chamamos **tampa do bolso** (ver figura 2.9).

Definição 2.1.11 *Um polígono simples,* P, *diz-se* ***estrelado*** *se existe pelo menos um ponto* $x \in P$ *tal que para todo ponto* $y \in P$ *o segmento* $[xy] \subset P$.

Ou seja, recorrendo ao conceito de visibilidade, um polígono é estrelado se existir pelo menos um ponto do qual se pode ver todo o polígono.

Definição 2.1.12 *O* ***núcleo*** *ou o* ***Kernel****,* $Ker(P)$*, de um polígono simples,* P*, é o conjunto de todos os pontos pertencentes a* P *que vêem todos os pontos de* P*, ou seja, pontos do interior de* P *e da fronteira de* P.
Em linguagem simbólica podemos escrever, $Ker(P) = \{x \in P \mid \forall\, y \in P, [xy] \subset P\}$.

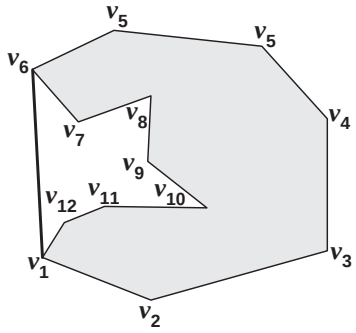


Figura 2.9: O bolso está limitado pelos vértices v_1, v_6, v_7, v_8, v_9, v_{10}, v_{11} e v_{12}. A tampa é o segmento entre os vértices v_1 e v_6.

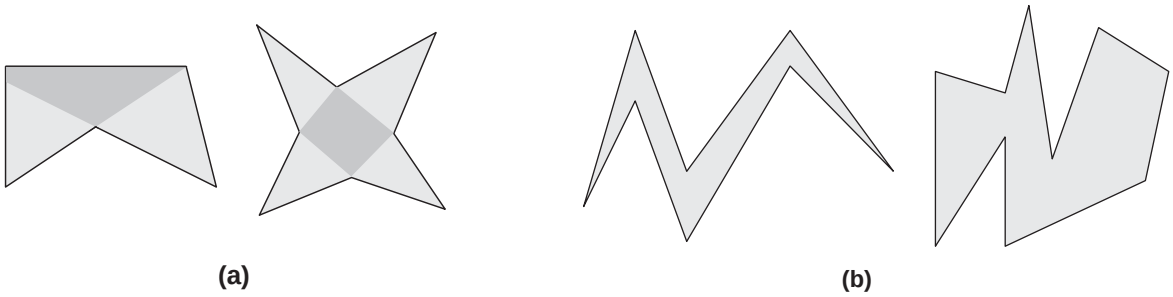


Figura 2.10: (a) Polígonos estrelados e os seus respectivos núcleos; (b) Polígonos não estrelados.

A definição 2.1.11 pode ser escrita de um modo diferente, recorrendo ao conceito de núcleo:

Definição 2.1.13 *Um polígono simples, P, é estrelado se o seu núcleo for diferente do vazio, isto é, se $Ker(P) \neq \emptyset$.*

Teorema 2.1.2 *O núcleo de um polígono é um conjunto convexo.*

Prova: A prova é evidente, por definição de $Ker(P)$, pois um conjunto X é convexo se $\forall x, y \in X \Rightarrow [xy] \subset P$. ■

Teorema 2.1.3 *Um polígono simples, P, é convexo se e só se $Ker(P) = P$.*

Prova:

($\Rightarrow$)

- $Ker(P) \subset P$

Seja $x \in Ker(P)$, obviamente que $x \in P$, pois por definição de $Ker(P)$, todos os pontos de $Ker(P) \in P$.

- $Ker(P) \supset P$

Como P é convexo, qualquer que seja $x \in P$, x vê qualquer ponto de P, logo $x \in Ker(P)$.

($\Leftarrow$)

Se $Ker(P) = P$ então pelo teorema 2.1.2, P é convexo. ■

Teorema 2.1.4 (Teorema de Krasnosel´ski) *Um conjunto S de $\mathbb{R}^2$ é estrelado se e só se qualquer terno de pontos pertencentes a S é visto por pelo menos um ponto de S, isto é, S é estrelado se e só se $\forall\, x, y, z \in S\,,\ \exists\, w \in S \mid w$ vê $x, y,$ e z.*

Prova: A prova do teorema de Krasnosel´ski pode ser obtida com a ajuda de um outro teorema: o Teorema de Helly. Este teorema afirma que se a intersecção dos conjuntos convexos $Q_1, Q_2, \ldots, Q_s$ do plano é vazia, então três destes conjuntos não têm nenhum ponto em comum. Consultar [61] para a prova completa. ■

Teorema 2.1.5 *Um polígono simples P é estrelado se e só se qualquer terno de vértices convexos é visto por pelo menos um ponto de P.*

A prova deste teorema deduz-se facilmente da prova do teorema anterior.

Definição 2.1.14 *Um polígono simples, P, diz-se* ***ortogonal*** *se todas as suas arestas forem paralelas ou ortogonais ao sistema de eixos coordenados.*

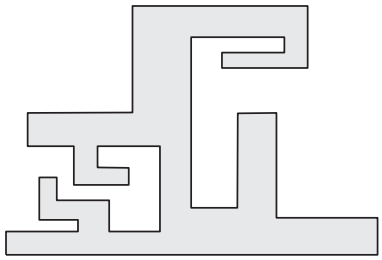


Figura 2.11: Polígono ortogonal.

O resultado seguinte, deve-se a J. O´Rourke [78].

Lema 2.1.2 *Seja P um polígono ortogonal com n vértices, r dos quais reflexos, então $r = \frac{n-4}{2}$.*

Prova: Como P tem n vértices, a soma da amplitudes dos seus ângulos internos é $(n-2)\pi$. Notemos que todos os ângulos internos de P têm amplitude $\frac{\pi}{2}$ ou $\frac{3\pi}{2}$, dependendo se é um vértice convexo ou reflexo, respectivamente. Assim, P tem r vértices reflexos e $n-r$ vértices convexos e ter-se-á:

$$(n-r)(\frac{\pi}{2}) + r\frac{3\pi}{2} = (n-2)\pi$$

Resolvendo em ordem a r teremos o resultado pretendido. ■

Uma **cadeia poligonal** diz-se **monótona** em relação a uma recta l, se qualquer recta ortogonal a l, intersecta a cadeia somente num vértice ou numa aresta.

Definição 2.1.15 *Um polígono simples, P, diz-se **monótono** em relação a uma recta l, se a fronteira de P se pode decompor em duas cadeias poligonais monótonas relativamente a l.*

Podemos, sem recorrer ao conceito de cadeia poligonal monótona, dar uma outra definição para polígono monótono.

Definição 2.1.16 *Um polígono simples, P, diz-se **monótono** em relação a uma recta l se a intersecção de qualquer recta ortogonal a l com o polígono, for um segmento de recta, um ponto ou vazia.*

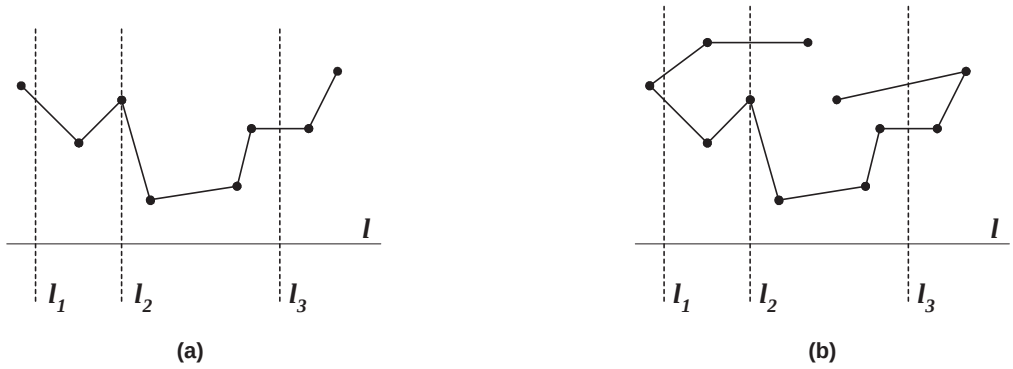


Figura 2.12: (a) Cadeia poligonal monótona em relação a l; (b) Cadeia poligonal não monótona em relação a l.

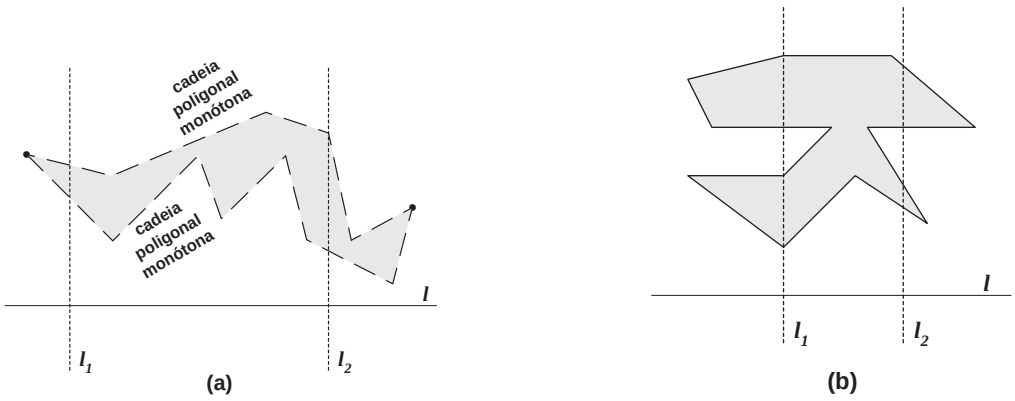


Figura 2.13: (a) Polígono monótono em relação a l; (b) Polígono não monótono em relação a l.

Um polígono monótono em relação ao eixo dos xx´s e em relação ao eixo dos yy´s diz-se **x-monótono** (figura 2.13 (a)) e **y-monótono**, respectivamente. Embora nas definições 2.1.15 e 2.1.16 se use a monotonia relativamente a uma qualquer recta, na prática a monotonia é usada apenas em relação aos eixos coordenados. A classificação de polígono monótono que é feita na secção 2.2 refere-se a uma monotonia relativamente aos eixos coordenados.

Definição 2.1.17 *Um polígono simples, P, diz-se* ***unimodal*** *se para cada ponto x pertencente à fronteira de P, se traçarmos segmentos com origem em x, passando pelo*

interior de P e fim nos seus vértices, obtivermos uma monotonia crescente até ao vértice mais distante de x, seguida de uma monotonia decrescente até ao vértice mais próximo de x.

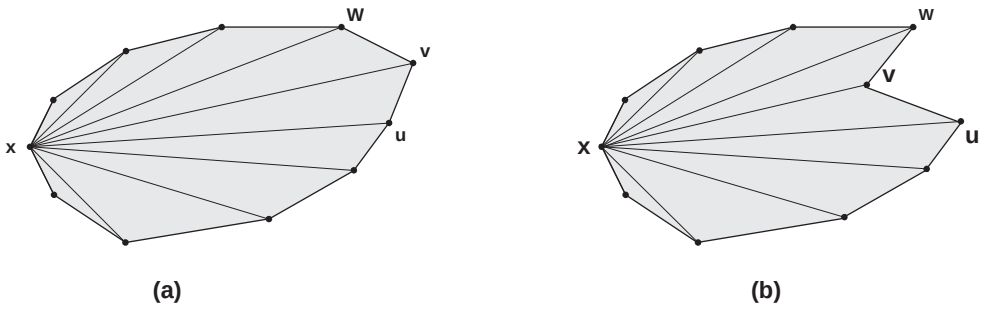


Figura 2.14: (a) Polígono unimodal em ordem a x; (b) Polígono não unimodal.

Definição 2.1.18 *Um vértice v_i de um polígono é uma **orelha** se o triângulo formado pelos vértices $[v_{i-1}, v_i, v_{i+1}]$ ($\triangle[v_{i-1}, v_i, v_{i+1}]$) pertence totalmente ao interior de P (ver figura 2.15 (a)).*

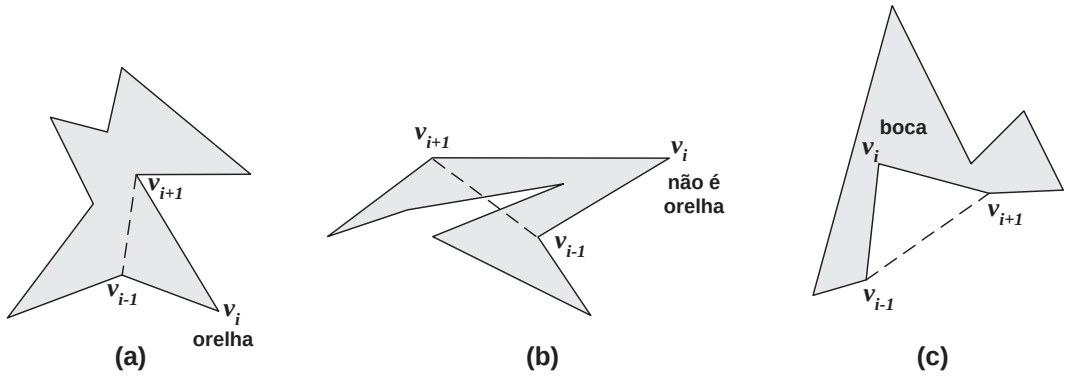


Figura 2.15: (a) O vértice v_i é uma orelha. (b) O vértice v_i não é orelha. (c) O vértice v_i é uma boca.

Dizemos que duas **orelhas** são **não coincidentes** se

$$\triangle[v_{i-1}, v_i, v_{i+1}] \cap \triangle[v_{j-1}, v_j, v_{j+1}] = \emptyset.$$

Definição 2.1.19 *Um vértice v de um polígono é uma* ***boca*** $\triangle[v_{i-1}, v, v_{i+1}]$ *pertence totalmente ao $ext(P)$ (ver figura 2.15 (c)).*

Definição 2.1.20 *Um polígono simples, P, diz-se* ***antropomórfico*** *se contém exactamente duas* orelhas *e uma* boca.

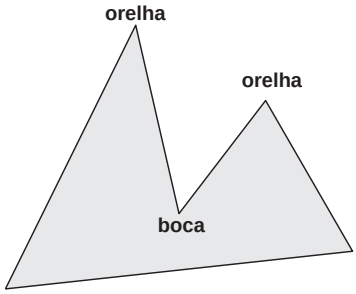


Figura 2.16: Polígono antropomórfico.

Teorema 2.1.6 (Teorema das duas orelhas) *Todos os polígonos simples com pelo menos quatro vértices têm pelos menos duas* orelhas *que não se sobrepõem.*

Prova: A prova deste teorema é feita por indução (estratégia proposta por G.H. Meisters) no número de vértices de P.

Passo básico:

$n = 4$

Quando $n = 4$, temos um quadrilátero que pode ser dividido em dois triângulos que serão as orelhas de P.

Hipótese de indução:

Qualquer polígono simples com menos do que n vértices (e com pelo menos quatro) tem duas orelhas.

Passo indutivo:

Sejam P um polígono com n vértices e r um vértice reflexo de P. Assim, fica eliminado o caso em que o triângulo pertence ao exterior de P). Sejam v_1 e v_2 dois vértices vizinhos. Teremos que considerar, obviamente, dois casos: ou r pertence à orelha ou não pertence.

Caso 1: Caso em que r pertence à orelha. Remove-se esta orelha de P, adicionando a aresta $[v_1, v_2]$ às outras arestas e obtemos, assim, um novo polígono simples. Este novo polígono tem $n-1$ vértices e terá duas orelhas excepto se for um triângulo, que terá somente uma orelha. Da hipótese de indução, temos que P tem duas orelhas.

Caso 2: No caso em que r não pertence à orelha, existe pelo menos um vértice no triângulo $[v_1, r, v_2]$. Tracemos uma paralela a $[v_1, v_2]$, partindo de v_1 até atingir o vértice antes de r. Chamemos-lhe v. Como não existem vértices mais próximos de r que o vértice v, então o segmento $[r, v]$ pertence ao interior de P. Este segmento divide P em dois polígonos. Chamemos-lhes $P_{esquerdo}$ e $P_{direito}$, onde $P_{esquerdo}$ é a parte do polígono que está à esquerda de $[r, v]$ e $P_{direito}$ é a outra parte. Os polígonos $P_{esquerdo}$ e $P_{direito}$ têm ambos menos do que n vértices, tendo, portanto, duas orelhas (hipótese de indução).

Teremos agora que mostrar que isto implica que P tem duas orelhas.

Pode suceder que ou $P_{esquerdo}$ ou $P_{direito}$ seja, um deles, um triângulo. Consideremos que o triângulo é o $P_{direito}$. Então $P_{direito}$ é uma orelha de P e $P_{esquerdo}$ tem duas orelhas. Seguramente que a nenhuma delas pertencem os vértices r ou v. Esta orelha é a segunda orelha de P e, então, P tem duas orelhas.

Pode ainda acontecer que nem $P_{esquerdo}$ ou $P_{direito}$ seja um triângulo e, neste caso, pela mesma razão explicitada anteriormente, cada polígono $P_{esquerdo}$ ou $P_{direito}$ tem pelo menos uma orelha, à qual não pertence nem o vértice r nem o vértice v. Estas duas orelhas são, então, as duas orelhas de P. ■

Teorema 2.1.7 (Teorema da boca) *Exeptuando os polígonos simples convexos, todos os polígonos simples têm pelo menos uma* boca.

Prova: Contruamos o invólucro convexo de P, $CH(P)$. Como P, por hipótese, é não convexo, há arestas pertencentes à fronteira do invólucro convexo de P, cada qual formando a tampa de um bolso de $CH(P)$ (ver figura 2.17). Temos então que provar

que cada bolso tem uma boca. Seja K_{ij} o bolso de $CH(P)$, determinado pelos vértices v_i e v_j de P. Obviamente que $K_{ij} = [v_i, v_{i+1}, ..., v_j] \cup [v_j, v_i]$ forma um polígono simples. Pelo teorema 2.1.6 (Teorema das duas orelhas), K_{ij} tem duas orelhas que, como não se sobrepõem, não podem ocorrer em v_i e v_j. Portanto, pelo menos uma orelha, deve-se encontrar no vértice v_k, $i < k < j$. Claramente que uma orelha para K_{ij} é uma boca para P. ■

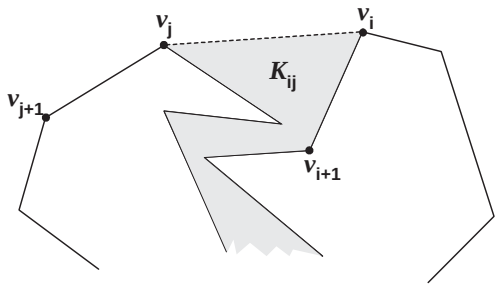


Figura 2.17: Ilustração da prova do teorema 2.1.7.

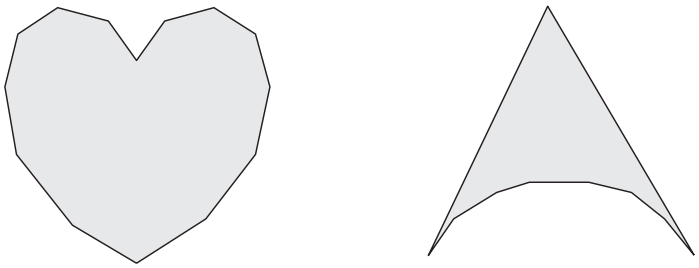


Figura 2.18: (a) Um polígono com apenas uma boca e várias orelhas. (b) Um polígono com duas orelhas e várias bocas.

Definição 2.1.21 *Um polígono simples, P, diz-se* ***visível do exterior****, que se abreviará por* ***VE****, se para todo ponto $x \in fr(P)$, existir uma semi-recta, l, com origem em x tal que $l \cap int(P) = \emptyset$ (ver figura 2.19 (a)).*

Definição 2.1.22 *Um polígono simples, P, diz-se* ***visível desde uma aresta****, que se abreviará por* ***VDA****, se existir em P uma aresta tal que para cada ponto y pertencente a*

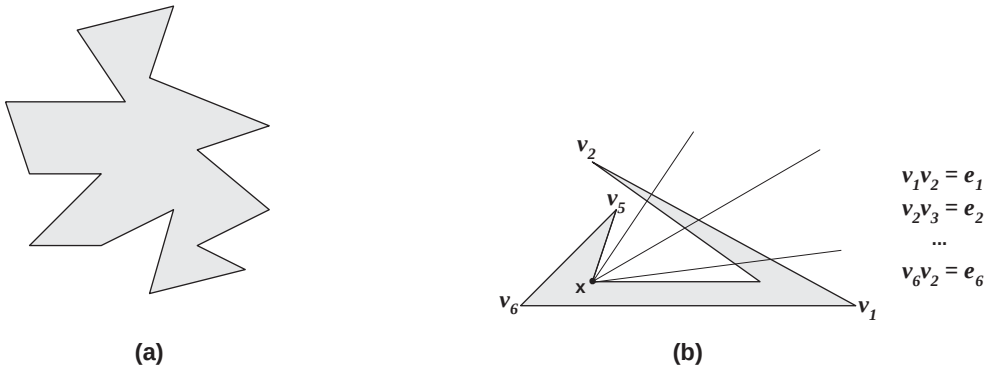


Figura 2.19: (a) Polígono VE; (b) Polígono não VE.

P, existe um ponto x pertencente à aresta, tal que o segmento $[xy]$ pertence ao interior de P, ou seja, para cada ponto y de P existe um ponto x da aresta que vê y, ou seja, tal que $[xy] \in int(P)$.

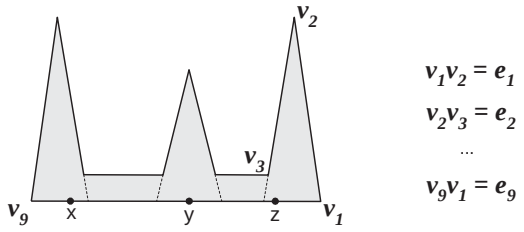


Figura 2.20: Todo o ponto de P é visível desde algum ponto da aresta e_9, em particular, desde os pontos x, y, e z.

Definição 2.1.23 *Um ponto y diz-se que tem **visibilidade fraca de uma aresta** se existir um ponto x nessa aresta, tal que y é visível de x.*

Todos os pontos do polígono da figura 2.20 têm visibilidade fraca da aresta e_9 como facilmente se pode verificar.

Definição 2.1.24 *Um polígono simples, P, diz-se **completamente visível desde uma aresta**, que se abreviará por **CVA**, se existir em P uma aresta tal que, para*

cada ponto y pertencente a P e para cada ponto x pertencente à aresta, o segmento $[xy] \in int(P)$.

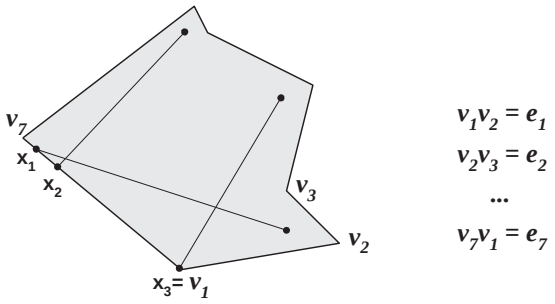


Figura 2.21: Polígono CVA relativamente a e_7.

O polígono da figura 2.21 é CVA relativamente a e_7, pois qualquer ponto, x_i, de e_7 vê todo o polígono ($i = 1, 2, 3, ...$). No entanto, este polígono não é CVA relativamente a e_2.

Como facilmente se pode constatar, qualquer polígono simples, pode ser dividido em duas cadeias poligonais. Uma **cadeia poligonal** diz-se **convexa** se todos os ângulos, que pertencem ao interior do polígono, forem convexos. Caso contrário a **cadeia poligonal** diz-se **côncava**.

Definição 2.1.25 *Um polígono simples, P, diz-se **polígono estrada** relativamente a dois determinados vértices, se a sua fronteira pode ser dividida, por esses vértices, em duas cadeias poligonais, tal que os pontos de cada uma das cadeias têm visibilidade fraca (ver figura 2.22 (a)).*

Notemos que o polígono da figura 2.22 (b) é não estrada relativamente a outros pares de vértices, por exemplo, v_1 e v_8.

Definição 2.1.26 *Um polígono simples, P, diz-se **polígono em espiral** se a fronteira pode ser dividida numa cadeia convexa e numa cadeia côncava.*

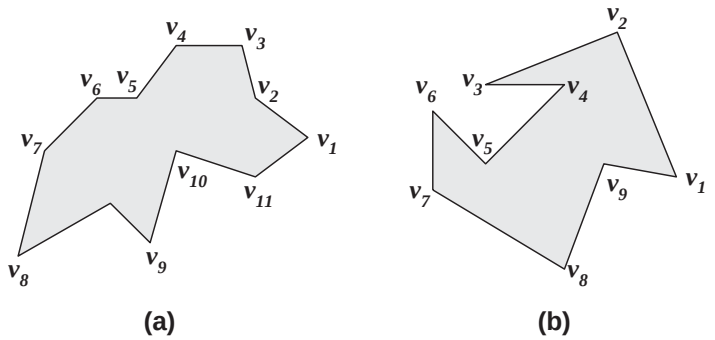


Figura 2.22: (a) Polígono estrada relativamente aos vértices v_1 e v_8; (b) Polígono não estrada relativamente aos vértices v_1 e v_7.

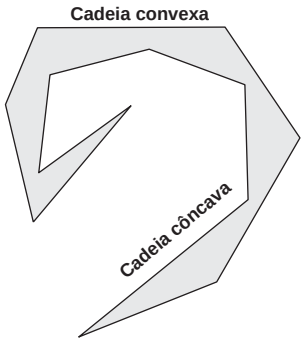
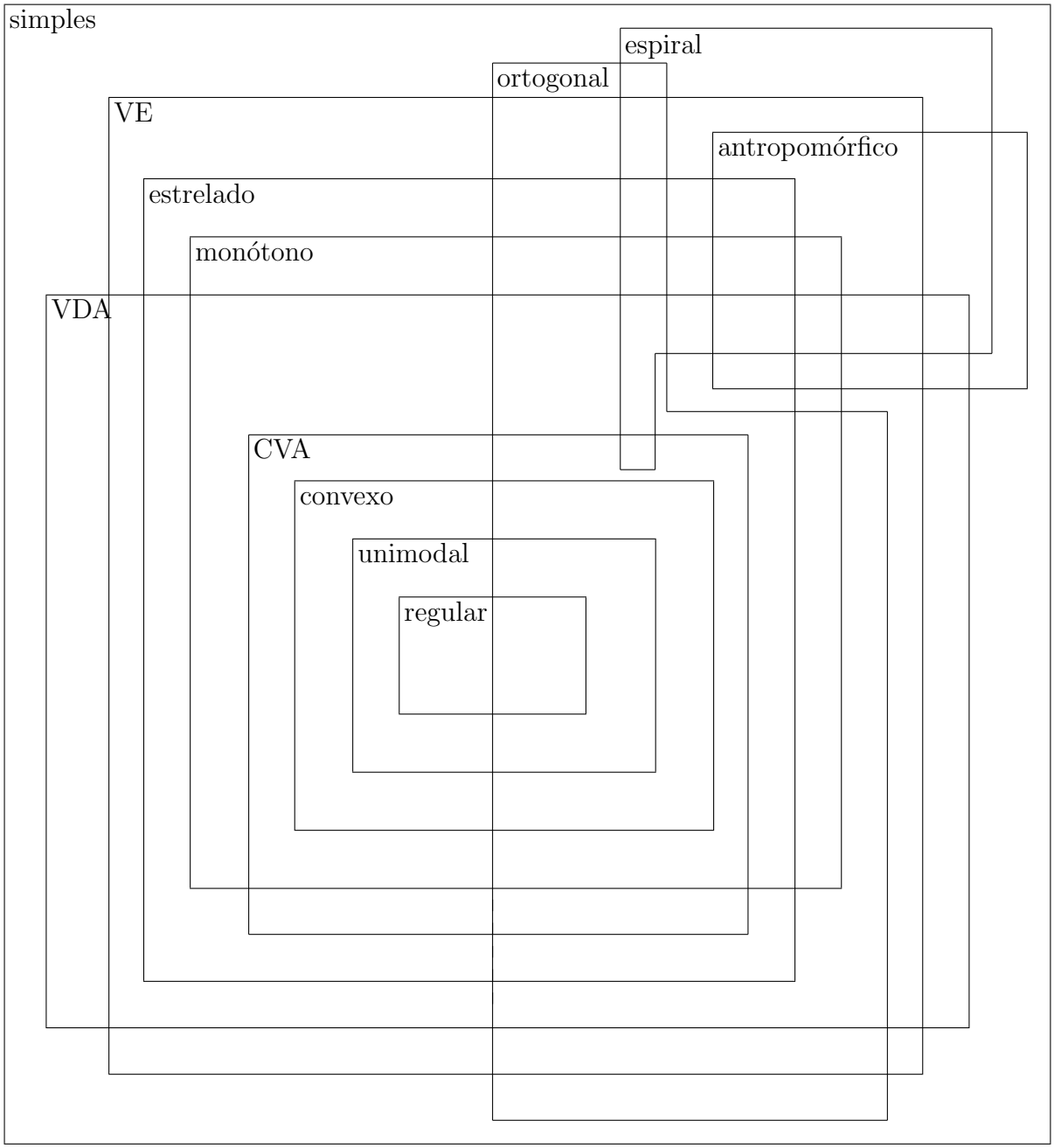


Figura 2.23: Polígono em espiral.

2.2 Uma classificação hierárquica de polígonos simples

Após a definição de vários polígonos simples na secção anterior, fazemos nesta secção, uma classificação dos polígonos descritos. Apenas o polígono estrada não será incluído na classificação, devido à sua particularidade de depender sempre dos vértices escolhidos. Por outro lado, embora não tenhamos definido na secção anterior, iremos considerar na hierarquização os polígonos regulares com o número de vértices superiores a 3, que são todo aqueles que têm arestas e ângulos iguais.


simples
espiral
ortogonal
VE
antropomórfico
estrelado
monótono
VDA
CVA
convexo
unimodal
regular

Através da hierarquização feita podemos constatar que:

	reg	uni	cvx	CVA	mtn	est	VE	VDA	atp	esp	ort	smp
reg	$\times$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\cap = \emptyset$	$\cap = \emptyset$	$\cap \not\subseteq$	$\subset$
uni	$\supset$	$\times$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\cap = \emptyset$	$\cap = \emptyset$	$\cap \not\subseteq$	$\subset$
cvx	$\supset$	$\supset$	$\times$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\cap = \emptyset$	$\cap = \emptyset$	$\cap \not\subseteq$	$\subset$
CVA	$\supset$	$\supset$	$\supset$	$\times$	$\cap \not\subseteq$	$\subset$	$\subset$	$\subset$	$\cap = \emptyset$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\subset$
mtn	$\supset$	$\supset$	$\supset$	$\cap \not\subseteq$	$\times$	$\cap \not\subseteq$	$\subset$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\subset$
est	$\supset$	$\supset$	$\supset$	$\supset$	$\cap \not\subseteq$	$\times$	$\subset$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\subset$
VE	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\times$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\subset$
VDA	$\supset$	$\supset$	$\supset$	$\supset$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\times$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\subset$
atp	$\cap = \emptyset$	$\cap = \emptyset$	$\cap = \emptyset$	$\cap = \emptyset$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\times$	$\cap \not\subseteq$	$\cap = \emptyset$	$\subset$
esp	$\cap = \emptyset$	$\cap = \emptyset$	$\cap = \emptyset$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\subset$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\times$	$\cap = \emptyset$	$\subset$
ort	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap \not\subseteq$	$\cap = \emptyset$	$\cap = \emptyset$	$\times$	$\subset$
smp	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\supset$	$\times$

Note-se que estamos somente a analisar polígonos simples, portanto de acordo com a definição 2.1.1. Podemos salientar, por exemplo, alguns pontos importantes:

- Um polígono que seja regular é unimodal, convexo, CVA, VDA, monótono, estrelado e VE mas não é antropomórfico nem espiral.
- Um polígono unimodal é convexo, CVA, VDA, monótono, estrelado e VE, mas não é antropomórfico nem espiral.
- Um polígono convexo é CVA, VDA, monótono, estrelado e VE, mas não é antropomórfico nem espiral.
- Um polígono CVA é VDA, VE e estrelado mas não antropomórfico.
- Um polígono monótono ou estrelado é VE.
- Um polígono ortogonal não é antropomórfico.
- Um polígono antropomórfico não pode ser nem CVA, nem convexo, nem unimodal, nem ortogonal.

Seguem-se alguns exemplos:

1. O polígono da figura 2.24 é monótono relativamente a qualquer um dos eixos coordenados e VDA relativamente às arestas e_5 e e_6.

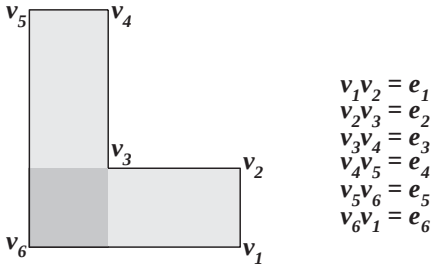


Figura 2.24: Polígono ortogonal, VDA, não CVA e estrelado.

2. O polígono da figura 2.25 é não monótono em relação ao eixo dos $yy's$, mas já é monótono relativamente ao eixo dos $xx's$. É VDA relativamente às arestas e_1 e e_2.

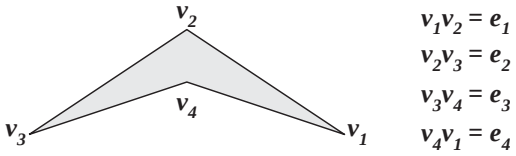


Figura 2.25: Polígono espiral, antropomórfico, VDA e não CVA.

3. O polígono da figura 2.26 é espiral (podemos dividi-lo em duas cadeias - uma convexa e outra côncava - nos vértices v_3 e v_6), é antropomórfico, tem uma boca no vértice v_5 e exactamente duas orelhas, nos vértices v_3 e v_6 e é não VE, pois não é possível traçar segmentos de recta com origem em pontos da aresta v_5 sem intersectar o polígono.

4. O polígono da figura 2.27 é VDA pois para qualquer ponto do polígono existe sempre um ponto da aresta e_1 que se consegue ver; é não VE pois do vértice v_5 não se consegue traçar uma semi-recta sem intersectar o polígono e é não espiral

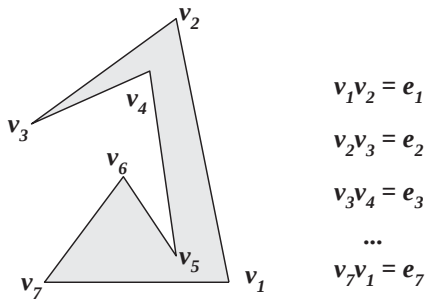


Figura 2.26: Polígono espiral, antropomórfico e não VE.

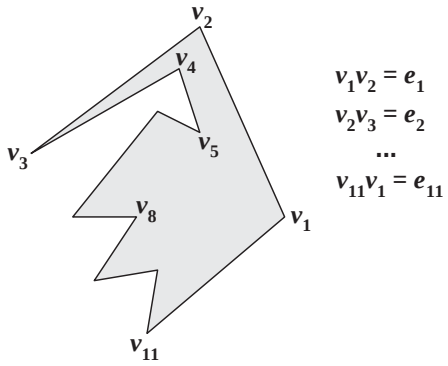


Figura 2.27: Polígono VDA, não VE e não espiral.

pois não existem dois vértices pelos quais possamos dividir o polígonos em duas cadeias poligonais, uma côncava e outra convexa.

5. O polígono da figura 2.28, também conhecido como **polígono pente**, é um exemplo de um polígono ortogonal, VDA da aresta e_1 e VE.

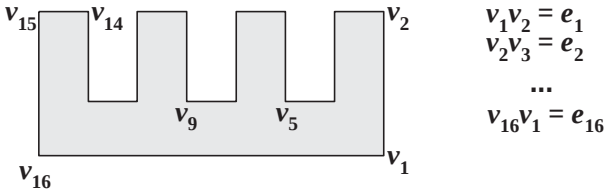


Figura 2.28: Polígono ortogonal, VDA e VE.

6. O polígono da figura 2.29 é somente simples e antropomórfico, tem uma boca no vértice v_9 e duas orelhas, nos vértices v_6 e v_{14}.

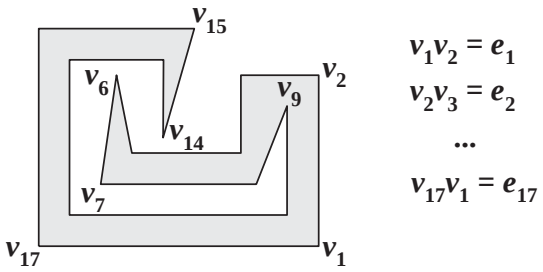


Figura 2.29: Polígono antropomórfico.

7. O polígono da figura 2.30 é VDA (aresta e_4), não monótono relativamente ao eixo dos yy´s, estrelado (núcleo a sombreado), VE e não antropomórfico, pois tem quatro orelhas (vértices v_1, v_2, v_4 e v_5).

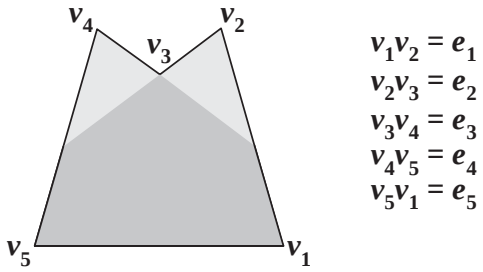


Figura 2.30: Polígono VDA, estrelado, VE, não y-monótono e não antropomórfico.

8. O polígono da figura 2.31 é ortogonal, espiral (pode-se dividir nos vértices v_5 e v_9), não VDA e não VE (vértice v_7, por exemplo).

9. O polígono da figura 2.32 é não VDA, não VE, não espiral, não monótono, não estrelado, não antropomórfico e não CVA.

10. Os únicos polígonos que são CVA, ortogonais e convexos são o quadrado e o rectângulo.

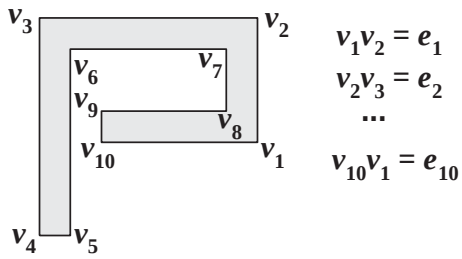


Figura 2.31: Polígono ortogonal, espiral, não VDA e não VE.

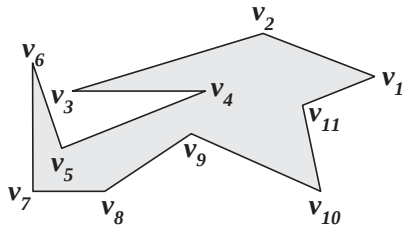


Figura 2.32: Polígono com ausência de características.

11. O polígono da figura 2.16 é um exemplo de um polígono antropomórfico, VDA mas não CVA.

12. O único polígono ortogonal, unimodal não regular é o rectângulo, sendo que, o único regular é o quadrado.

13. O polígono da figura 2.19 (b) é um exemplo de um polígono que não sendo VE, é VDA e é espiral (podemos dividi-lo em duas cadeias - uma convexa e outra côncava - nos vértices v_2 e v_5) .

Capítulo 3

Partição clássica de polígonos

A decomposição de um polígono, ou de qualquer outra região, em partes mais simples é útil em muitos problemas. Na maioria dos casos, essas partes mais simples são triângulos, no entanto, podem ser usadas outras formas, como, por exemplo, quadriláteros ou simplesmente peças convexas. A decomposição de polígonos é classificada de acordo como as suas componentes se interligam. Assim, uma decomposição diz-se que é uma **partição** se as componentes do subpolígono não se sobrepõem, excepto na sua fronteira. Se houver sobreposição de componentes, então dizemos que a decomposição é uma **cobertura**.

A partição de polígonos é usada frequentemente para modelar objectos em aplicações onde a geometria é importante. Existem muitas aplicações teóricas e práticas da partição de polígonos tendo estas sido objecto de vários estudos [18, 52, 78, 97, 109]. O reconhecimento de padrões é uma das áreas em que a decomposição de polígonos é usada como uma ferramenta [32, 81, 80, 33, 109]. As técnicas de reconhecimento de padrões, extraem informação de um objecto com o objectivo de o descrever, identificar e classificá-lo. Uma estratégia normal para reconhecer um dado objecto poligonal, é decompô-lo em componentes mais simples, identificando, então, as componentes interligando-as depois, e usar esta informação para determinar a forma do objecto [32, 80]. Existem muitas outras aplicações de decomposição de polígonos, tais como, compressão de dados [71], sistemas de bases de dados [68], processamento de imagem [76] e computação gráfica [102].

Em Geometria Computacional, os algoritmos para problemas de decomposição

de um qualquer polígono são mais complexos que os algoritmos para os mais restritos, como sejam os algoritmos de decomposição de polígonos convexos ou estrelados. A estratégia para resolver alguns destes problemas, em polígonos quaisquer, é decompô-lo em componentes mais simples, resolver o problema para cada uma das componentes, usando o algoritmo adequado e depois combinar as diversas soluções.

Existe uma grande variedade de classes de polígonos que são úteis para a decomposição de polígonos, são os casos dos convexos, estrelados, espirais, monótonos, triângulos, quadrados, rectângulos e trapézios. Esta decomposição em componentes mais simples, pode ser feita (ou não) com a introdução de vértices adicionais, aos quais chamamos **pontos de Steiner** [40]. Apesar do uso de pontos de Steiner tornar a decomposição do polígono mais complexa, reduz, na maioria dos casos, o número de componentes. A complexidade da decomposição de um algoritmo é analisada tendo em conta o número inicial de vértices do polígono e o número de vértices côncavos que se formam com os pontos de Steiner. Na maioria das aplicações, pretende-se que a decomposição seja minimal em algum sentido, por exemplo, em algumas aplicações procura-se decompor o polígono num número mínimo de componentes. Outras aplicações, usam uma decomposição que minimiza o comprimento total das arestas internas usadas para a decomposição. Pensa-se que o primeiro resultado, que minimizou este comprimento total, deve-se a Klincsek [57] que, em 1980, usou programação dinâmica para encontrar o comprimento total mínimo, numa triangulação de um polígono. O trabalho de Klincsek foi influente na medida em que inspirou outras soluções com programações dinâmicas em problemas de decomposição. Como no exemplo da figura 3.1, uma decomposição (partição) de comprimento mínimo 3.1(b) pode ser diferente de um número mínimo de decomposições 3.1 (a) para o mesmo tipo de componente.

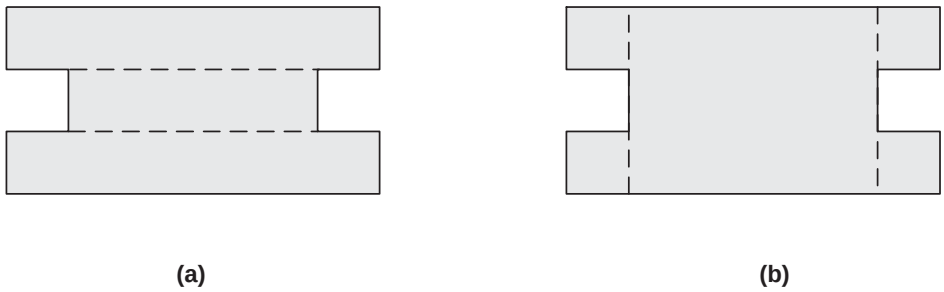


Figura 3.1: Duas partições com características diferentes.

A aplicação é que determina o tipo de subpolígono a usar quando se particiona um polígono em subpolígonos mais simples; por exemplo, o reconhecimento de padrões, usa subpolígonos convexos, espirais e estrelados na decomposição [32, 81, 33, 96, 109]; as aplicações VSLI usam trapézios [6].

É de total interesse, também, (e, por isso, este tema será tratado na secção seguinte) o desenvolvimento de algoritmos que particionem um polígono em triângulos, muito usado, por exemplo, em problemas do tipo da Galeria de Arte [78].

3.1 Triangulação de polígonos simples

A triangulação de polígonos simples é um problema clássico da Geometria Computacional e um dos primeiros a ser estudado nesta área. O problema da triangulação de polígonos, pode ser formulado da seguinte maneira: dado um polígono P, com n vértices, encontrar diagonais que particionem o polígono em triângulos [95] (ver figura 3.2 (a)). Como se pode observar na figura 3.2 (b) a triangulação de um polígono pode não ser única.

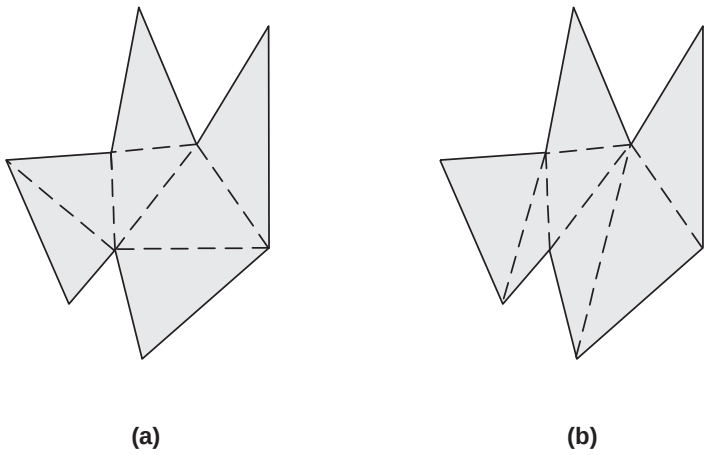


Figura 3.2: Duas triangulações distintas do mesmo polígono.

O primeiro algoritmo para construir uma triangulação de um polígono, foi proposto por Lennes [3.1.3], em 1911, usando um método recursivo de inserção de diagonais, cuja complexidade é de $O(n^2)$.

Provavelmente, o resultado mais importante em triangulação foi um algoritmo criado por Garey, Johnson, Preparata e Tarjan [38], o primeiro a quebrar o tempo $O(n^2)$ e que executa uma triangulação em tempo $O(n\,log\,n)$. Este era precisamente o tempo que demorava o primeiro passo, que era a decomposição do polígono em subpolígonos y-monótonos. Este algoritmo de decomposição em polígonos y-monótonos, foi proposto por Preparata [65]. Garey et al. aplicaram, depois, um algoritmo para triangular um polígono monótono em tempo linear, $O(n)$. Aplicando este algoritmo a cada subpolígono y-monótono, completa-se o algoritmo inicial. Já há contudo, algoritmos mais eficientes, no entanto, este é de fácil implementação sendo usado com bastante frequência na prática. Uma abordagem completamente diferente à usada por Garey et al. foi proposta por Chazelle [14], que usou a técnica "dividir para conquistar". No entanto, a complexidade do algoritmo continuou a ser $O(n\,log\,n)$.

Um outro algoritmo para triangular polígonos simples, também com tempo de execução $O(n\,log\,n)$, foi apresentado por Mehlhorn [73]. Este algoritmo baseia-se na ideia de varrimento (isto é, em modos gerais, atravessar o polígono da esquerda para a direita, usando uma linha vertical).

Os resultados obtidos, também neste campo da triangulação de polígonos, por Asano e Pinter [7] levou-os a deixar uma questão em aberto: "é possível triangular um polígono simples num tempo $o(n\,log\,n)$?" ($o(n\,log\,n)$ significa estritamente menor que $O(n\,log\,n)$).

Foram muitos os investigadores que já trabalharam neste problema. Uma aproximação foi feita encontrando classes de polígonos que podiam ser triangulados num tempo $O(n)$, como, por exemplo, as classes de polígonos monótonos [38, 104], estrelados [94, 111], visíveis desde uma aresta (VDA) [110], em espiral [32], antropomórficos [105], etc.. Esta classe foi apelidada de **polígonos com triangulação linear**.

Três das principais motivações para a procura de algoritmos $o(n\,log\,n)$ são as seguintes:

- extendendo sucessivamente esta classe de polígonos, pode-se, eventualmente, encontrar um algoritmo para qualquer polígono.
- Provavelmente, uma destas classes tem um algoritmo de subdivisão, como o de Lee e Preparata, de complexidade $O(n)$, podendo depois chegar-se a um algoritmo de tempo $O(n)$ para qualquer polígono.

- Algumas classes são interessantes por direito próprio e aplicações que usem essas classes podem beneficiar do algoritmo de ordem linear. Esta aproximação originou muitas classes de polígonos com triangulação linear [28, 36, 39, 43, 64, 110, 111], mas não proporcionou um avanço real para esta classe.

Outra aproximação para uma triangulação em tempo $o(n\,log\,n)$ foi encontrada usando algoritmos cujo tempo de execução se baseou nas características estruturais do polígono. Os mais notáveis foram os algoritmos de Hertel e Mehlhorn [46] e de Chazelle e Incerpi [19].

O algoritmo de Hertel e Mehlhorn tem complexidade $O(n + r\,log\,r)$, onde r representa o número de vértices côncavos (reflexos) e é tanto mais eficaz quanto menos vértices côncavos o polígono tiver. Foi um algoritmo criado usando a técnica de varrimento.

Já o algoritmo de Chazelle e Incerpi [19] tem complexidade $O(n\,log\,s)$, onde s representa a sinuosidade do polígono, sendo $s < n$. A sinuosidade é um parâmetro que nos *mede* as alterações na fronteira do polígono, isto é, o número de vezes que a fronteira do polígono alterna entre espirais completas de orientações contrárias. Consideremos o movimento de uma recta $L_{[v_i,v_{i-1}]}$ que percorre a aresta $[v_i, v_{i-1}]$, com $1 < i < n-1$. Cada vez que $L_{[v_i,v_{i-1}]}$ atinge a posição vertical, no sentido dos ponteiros do relógio acrescentamos ao "contador de sinuosidade"1. Caso a posição vertical seja atingida com um movimento contrário ao sentido dos ponteiros do relógio, tiramos 1 ao *contador*. $L_{[v_i,v_{i-1}]}$ diz-se que está em espiral (respectivamente em anti-espiral) se o "contador"nunca tiver sido decrementado (respectivamente aumentado) duas vezes seguidas. Desta forma, o polígono poderá ser facilmente particionado num tempo $O(n)$ em cadeias espirais e anti-espirais. Um exemplo de um polígono com sinuosidade 5 é mostrado na figura 3.3. Notemos que uma cadeia poligonal recomeça somente quando a cadeia anterior pára de ser espiral (ou anti-espiral). A sinuosidade de P é definida como sendo o número de cadeias poligonais assim obtidas.

Na prática s é um valor muito pequeno, mesmo em polígonos com uma forma "complicada". O algoritmo de Chazelle e Incerpi é, teoricamente, muito mais interessante que o algoritmo de Hertel e Mehlhorn por causa das implicações que tem na complexidade da triangulação nas conhecidas diferentes classes de polígonos. Como r, que representa o número de vértices côncavos (reflexos), é independente do facto do polígono ser monótono, estrelado, VDA, etc., o algoritmo de Hertel e Mehlhorn pode

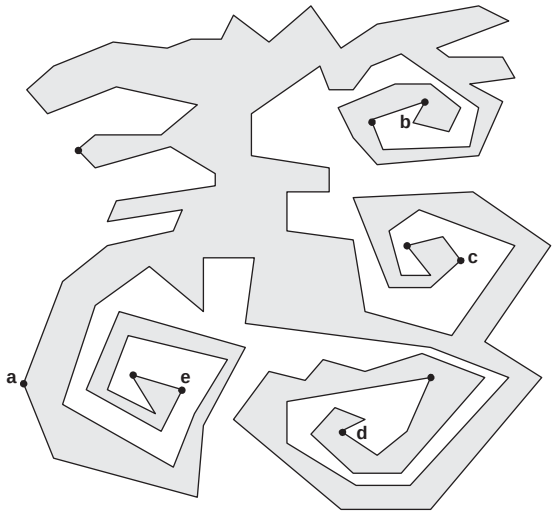


Figura 3.3: Polígono com sinuosidade 5. O início da verificação é no vértice a.

ser executado num tempo $O(n \log n)$ para estas classes de polígonos, para as quais são conhecidos os algoritmos com tempo linear. Por outro lado, os polígonos estrelados têm sinuosidade 1 e assim o algoritmo de Chazelle e Incerpi executa um tempo linear para estes algoritmos. Além disso, o algoritmo não faz uso do núcleo de P. Em [94] e [111] é necessário um ponto no núcleo do polígono e isto implica um esforço extra (apesar do tempo ser linear). Para uma abordagem completamente diferente usando um algoritmo extremamente simples para a triangulação de um polígono estrelado, não recorrendo ao seu núcleo ver [28] ou [29]. No entanto, a sinuosidade não é uma medida completamente satisfatória da complexidade da estrutura do polígono pois tem uma propriedade desconcertante que é a de poder variar dependendo da orientação do polígono. Por exemplo, consideremos o polígono VDA ilustrado na figura 3.4. A sinuosidade deste polígono é $O(n)$ e assim o algoritmo de Chazelle e Incerpi é executado num tempo $O(n \log n)$ visto que existe um algoritmo de tempo linear [110]. Além disso, fazendo uma rotação de 90º graus do polígono, a sinuosidade fica reduzida a $O(1)$. Isto representa uma variação na sinuosidade de P sem que tenha havido qualquer tipo de alteração no polígono (naturalmente que assumimos que a forma do polígono é invariante sob uma translação ou rotação).

Este algoritmo, em relação ao de Hertel e Mehlhorn, reflecte mais fielmente a

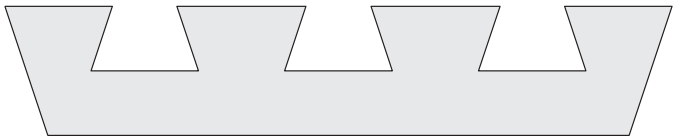


Figura 3.4: Polígono VDA com sinuosidade $O(n)$.

forma do polígono. Ao contrário de r, s tem a vantagem de que em muitas situações práticas é bastante pequeno ou então é uma constante, mesmo para polígonos muito *sinuosos*.

Já constatámos que Garey et al. [38] afirmaram que a decomposição em partes monótonas e a decomposição em triângulos são de complexidade linearmente equivalentes, ou seja, tendo uma decomposição, podemos encontrar a outra num tempo $O(n)$. Ora, uma terceira aproximação ao tempo $o(n \log n)$, foi encontrar outras decomposições que são linearmente equivalentes à triangulação. Fournier e Montuno mostraram-nos que a triangulação e a trapezoidação (entre outras decomposições) são de complexidade linearmente equivalentes [36]. Chazelle e Incerpi também provaram esta equivalência linear.

Tarjan e Van Wyk foram os primeiros a estabelecer um tempo $o(n \log n)$ para uma triangulação de um polígono [101], quebrando assim, a barreira $O(n \log n)$. O seu algoritmo, usava uma complicada e sofisticada estrutura de dados e era executado num tempo $O(n \log \log n)$. Actualmente, este algoritmo executa uma trapezoidação em vez de uma triangulação. Dois anos depois a mesma complexidade foi demonstrada, usando uma estrutura da dados simples, por Kirkpatrick, Klawe e Tarjan [54]. Entretanto, Clarkson, Tarjan e Van Wyk [22], Devillers [24] e Seidel [95] já haviam desenvolvido algoritmos aleatórios com um tempo $O(n \log^* n)$, mostrando assim que a técnica que usaram para os seus algoritmos (a aleatoriedade) era uma boa ferramenta no desenvolvimento de algoritmos mais rápidos. Estes algoritmos, não só eram mais rápidos que os $O(n \log \log n)$, mas também mais simples.

Um outro algoritmo com particular relevância, desenvolvido nos anos 90, baseado no método da pesquisa de Graham, foi proposto por Kong, Everett e Toussaint [59]. A pesquisa de Graham é uma técnica *backtracking* fundamental em Geometria Computacional que foi originalmente criada para executar o invólucro convexo de um

conjunto de pontos no plano [41] e tem, desde então, muitas aplicações em diferentes contextos. Em [59] é mostrado como a pesquisa de Graham é usada para triangular um polígono simples num tempo $O(rn)$ onde $r-1$ representa o número de vértices côncavos de P. Apesar de no pior caso, este algoritmo ser executado em $O(n^2)$ e, por esta razão, não tão assimptoticamente eficaz como o algoritmo de Hertel e Mehlhorn, ele é de muito mais fácil implementação e, como tal, muito mais interessante do ponto de vista prático.

Talvez o algoritmo de mais fácil implementação e com um rápido tempo de execução logo, em termos práticos, mais eficiente, tenha sido o proposto por Toussaint [107]. Este algoritmo tem complexidade $O(n\,(1+t_0))$, com $t_0 < n$. A quantidade t_0 representa o número de triângulos que após a triangulação não partilham arestas com o polígono e está relacionado com a complexidade da estrutura deste. Apesar de tudo, no pior dos casos o algoritmo é $O(n^2)$, mas para muitas classes de polígonos o algoritmo é executado num tempo muito próximo do linear. Este algoritmo além de ser muito simples, não necessita da utilização da ordenação nem uma estrutura de árvores balanceadas, o que em termos práticos são grandes vantagens. Em termos teóricos tem interesse, pois é o primeiro algoritmo de triangulação cuja complexidade computacional é uma função do parâmetro de saída, neste caso da quantidade de triângulos (t_0).

Finalmente, em 1991, Chazelle [16] resolveu a questão posta por Asano e Pinter, apresentando um algoritmo de complexidade $O(n)$ para a triangulação de polígonos. Em termos teóricos, esta descoberta, foi um importante avanço na teoria da triangulação. Contudo, este algoritmo é de difícil implementação, pelo que, em termos práticos, o avanço não foi significativo. O desenvolvimento de um algoritmo de triangulação simples de complexidade linear continua em aberto.

Sintetizemos, então, a evolução da complexidade dos tempos da triangulação de polígonos simples:

- 1911 Lennes: $O(n^2)$ pelo método recursivo da inserção de diagonais.
- 1975 Meister: $O(n^3)$ pela técnica do corte de orelhas (*ear-cutting*).
- 1978 Garey, Johnson, Preparata, Tarjan: $O(n\,log\,n)$ pela decomposição em componentes monótonas.
- 1982 Chazelle: $O(n\,log\,n)$ pela técnica de *dividir para conquistar*.

- 1983 Herbert e Mehlhorn: $O(n + r\,log\,r)$ onde r representa o número de vértices côncavos.
- 1983 Chazelle: $O(n\,log\,s)$ onde s é a sinuosidade de P.
- 1987 Tarjan e Van Wyk: $O(n\,log\,log\,n)$ usando uma estrutura de dados complexa.
- 1988 Toussaint: $O(n\,(1+t_0))$, via *sleeve*[1] *shearching*, onde t_0 representa o número de *triângulos livres* nos resultados da triangulação.
- 1989 Clarkson, Tarjan e Van Wyk: $O(n\,log^*n)$, onde log^*n é a iteração do logaritmo de n (ou seja, o número de vezes que usamos o logaritmo antes do resultado ser inferior a 1).
- 1990 ElGindy, Everett, Toussaint: encontrar uma orelha num tempo de $O(n)$ pela técnica *prune and search* implica que o algoritmo de Meister é dado no tempo de $O(n^2)$.
- 1991 Seidel: $O(n\,log^*n)$, pela técnica dos *trapézios aleatórios*.
- 1991 Chazelle: $O(n)$, muitas técnicas envolvidas - algoritmo de difícil implementação.

3.1.1 Teoria de Triangulações

Para se fazer a triangulação de um polígono, recorrendo a diagonais, é necessário fazer a sua partição em subpolígonos (triângulos) por meio de inserção de segmentos de recta (diagonais) que ligam vértices que não sejam adjacentes.

Definição 3.1.1 *Uma* ***diagonal*** *de um polígono* P *é um segmento de recta que liga dois vértices não adjacentes e que pertence totalmente ao interior de* P*, excepto os pontos de ligação, que pertencem à fronteira de* P*.*

Isto significa que uma diagonal *corta* um polígono simples em exactamente dois subpolígonos simples (figura 3.5(a), diagonal $[v_2v_4]$). Esta partição é sempre possível, pelo teorema 3.1.2.

[1]Uma *sleeve* é um polígono triangulado cuja árvore dual é uma cadeia.

Na figura 3.5 (a), o segmento que une os vértices v_2 e v_4 é uma diagonal. O segmento que une v_5 e v_{11} não é uma diagonal.

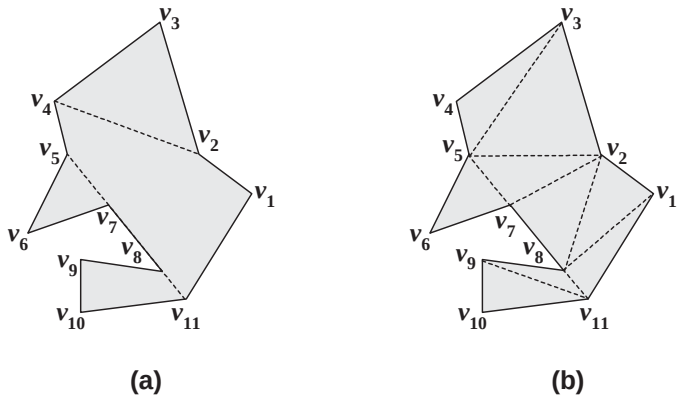


Figura 3.5: (a) Exemplo de uma diagonal e de uma não diagonal; (b) Triangulação de um polígono simples.

Lema 3.1.1 *Seja* $P = (v_0, v_1, ..., v_{n-1})$ *um polígono. Então* $s = [v_i v_j]$, $i \neq j$ *é uma diagonal de* P *se e somente se:*

1. *para cada aresta* e *de* P*, que não é incidente a* v_i *ou a* v_j*, temos que* $(s \cap e = \emptyset)$*; e*
2. $s \subset P$ *numa vizinhança de* v_i *ou de* v_j*.*

Prova: Para testarmos se um segmento $s = [v_i v_j]$ satisfaz a condição (1) do lema, basta aplicarmos o teste de intersecção[2] entre s e no máximo $n-4$ arestas de P. Para cada aresta e do polígono não incidente aos pontos extremos da diagonal s, temos que testar se e intersecta s. Quando a intersecção for detectada, sabemos que s não é uma diagonal. Se nenhuma aresta intersectar s, então s poderá ser uma diagonal. A razão pela qual não podemos tirar a conclusão imediata é porque é possível que uma das arestas incidente a um ponto extremo de s possa ser colinear com s e isso pode não ser detectado. Este teste de intersecção pode ser realizado em tempo $O(n)$.

Antes de provarmos o ponto 2, teremos que abrir um parêntesis para explicar de

[2]Consultar o código em Computacional Geometry in C, Cambridge University Press, First Edition, 1994.

que forma iremos mostrar que um ponto está ou não à esquerda de uma recta orientada. Uma recta orientada é determinada por um vector $\overrightarrow{ab}$, onde a e b são pontos. Se um ponto c está à esquerda dessa recta orientada então o terno (a, b, c) forma um circuito no sentido anti-horário. Um ponto c está à esquerda da recta orientada determinada pelo vector $\overrightarrow{ab}$ se e somente se a orientação do triângulo $\triangle(a, b, c)$ é positiva. O predicado *LeftOn* devolve verdadeiro se o ponto c estiver à esquerda ou sobre a recta orientada determinada pelo vector $\overrightarrow{ab}$. O predicado $Left$ determina se um ponto está à esquerda ou à direita de uma recta orientada. Este predicado recebe como parâmetros três pontos e é verdadeiro se e somente se o ponto c está à esquerda da recta orientada determinada pelo vector $\overrightarrow{ab}$.

Para determinarmos, então, se $s = [v_i v_j]$ está no interior do polígono numa vizinhança de, por exemplo, v_i (ponto (2)) temos de considerar dois casos:

1. v_i é vértice convexo. O vértice v_i é convexo se o predicado

   $$LeftOn(v_{i-1}, v_i, v_{i+1})$$

   é verdadeiro (ver figura 3.6). Neste caso, o segmento $s = [v_i v_j]$ está no interior de P na vizinhança de v_i se e somente se ambos os predicados $Left(v_i, v_j, v_{i-1})$ e $Left(v_j, v_i, v_{i+1})$ são verdadeiros, ou seja,

   $$Left(v_i, v_j, v_{i-1}) \wedge Left(v_j, v_i, v_{i+1})$$

   é verdadeiro.

2. v_i é vértice reflexo. O vértice v_i é reflexo se ele não for convexo, ou seja, o vértice v_i é reflexo se o predicado

   $$\sim LeftOn(v_{i-1}, v_i, v_{i+1})$$

   é verdadeiro (ver figura 3.7). Neste caso, o segmento $s = [v_i v_j]$ está no interior de P na vizinhança de v_i se e somente se o predicado

   $$\sim (Left(v_i, v_j, v_{i-1}) \wedge Left(v_j, v_i, v_{i+1}))$$

   é verdadeiro. ■

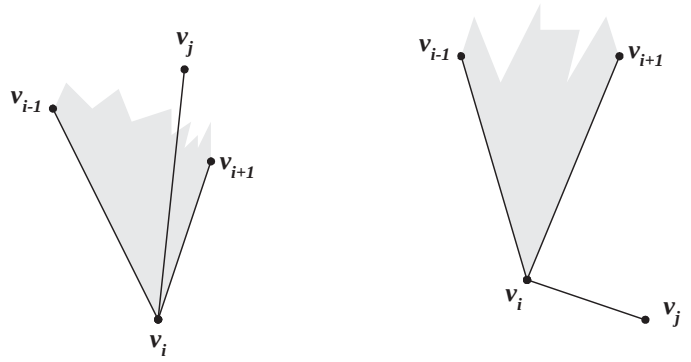


Figura 3.6: Possíveis situações quando v_i é convexo.

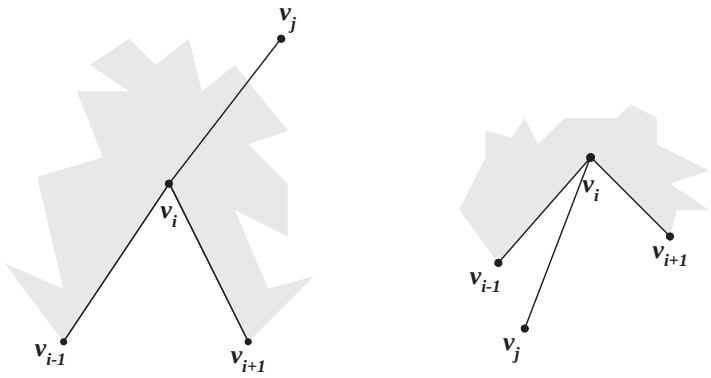


Figura 3.7: Possíveis situações quando v_i é reflexo.

O seguinte algoritmo testa se um dado segmento é uma diagonal:

Algoritmo *Diagonal*(P, v_i, v_j)
Entrada: um polígono $P = (v_0, ..., v_{n-1})$ e vértices v_i e v_j, $v_i \neq v_j$.
Saída: TRUE se $s := [v_i v_j]$ é uma diagonal de P e FALSE, caso contrário.

1. **for** toda a aresta e de P não incidente a v_i ou v_j **do**
2. **if** e e s se intersectam **then return** FALSE;
3. **if** v_i é convexo ($LeftOn(v_{i-1}, v_i, v_{i+1})$) **then**
4. **return** $Left(v_i, v_j, v_{i-1}) \wedge Left(v_j, v_i, v_{i+1})$;

5. **else**

6. **return** $\sim (Left(v_i, v_j, v_{i-1}) \wedge Left(v_j, v_i, v_{i+1}))$.

Este algoritmo tem complexidade $O(n)$, determinada pelo passo 1. Podemos assim, enunciar o seguinte teorema:

Teorema 3.1.1 *Seja P um polígono com n vértices e sejam u e v vértices de P. Então determinar se o segmento $[uv]$ é diagonal leva tempo $O(n)$.*

Lema 3.1.2 (Meister [74]) *Qualquer polígono simples P com mais do que três vértices tem uma diagonal.*

Prova: Seja P um polígono com mais do que três vértices e seja v um vértice estritamente convexo de P. Sejam u e w vértices adjacentes a v. Se $[uw]$ é uma diagonal do polígono então não há nada a provar. Suponhamos, então, que $[uw]$ não é uma diagonal de P, ou seja:

- ou $[uw] \nsubseteq P$
- ou $[uw] \subset P$ e $[uw] \cap fr(P) \nsubseteq \{u, w\}$

Como o número de vértices é superior a 3, então o triângulo de vértices v, u, w, denotado por $\triangle(v, u, w)$, contém pelo menos um vértice de P distinto de v, u e w. Seja t um vértice de P em $\triangle(v, u, w)$ mais próximo de v, onde a distância é medida ortogonalmente à recta passando pelo segmento $[uw]$. Logo, t é o primeiro vértice de P atingido quando movemos a recta, l, paralela a $[uw]$ de v na direcção de $[uw]$ (ver figura 3.8).

Afirmamos que $[vt]$ é uma diagonal de P. De facto, seja L a recta passando por t e paralela ao segmento $[uw]$. Notemos que a intersecção do semiplano determinado por L contendo o vértice v com o triângulo $\triangle(v, u, w)$ é um triângulo que não contém nenhum ponto de $fr(P)$ no seu interior. Logo, o vértice v vê claramente t. Portanto, $[vt]$ é uma diagonal de P. ■

Do teorema 3.1.1 e do lema 3.1.2 obtemos o seguinte teorema:

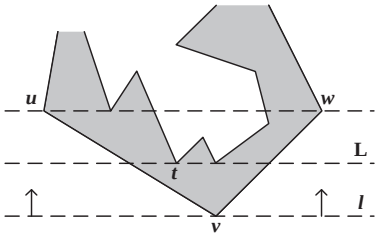


Figura 3.8: Ilustração da prova do Lema 3.1.2.

Teorema 3.1.2 *Qualquer polígono simples, P, com mais do que três vértices pode ser particionado em dois subpolígonos num tempo $O(n)$ por inserção de alguma diagonal de P.*

Teorema 3.1.3 *Qualquer polígono simples P admite uma triangulação.*

Prova: A prova é feita por indução no número n de vértices do polígono P. Se $n = 3$ o polígono é um triângulo e não há nada a provar.

Suponhamos que P não é um triângulo, ou seja, $n \geq 4$ então existe uma diagonal, d, pelo lema 3.1.2 que particiona P em dois subpolígonos com menos vértices que P, tendo cada um, d como aresta. Aplicando a hipótese de indução, ambos os subpolígonos admitem uma triangulação. Logo, combinando as triangulações de cada um dos polígonos e d obtemos uma triangulação de P. ■

Apesar de uma triangulação não ser única, o número de triângulos é sempre igual.

Teorema 3.1.4 *Qualquer triangulação de um polígono simples P com n vértices tem exactamente $n - 2$ triângulos.*

Prova: Consideremos uma diagonal qualquer. Esta diagonal divide P em dois subpolígonos, P_1 e P_2 com n_1 e n_2 vértices, respectivamente. Cada vértice de P pertence exactamente a um dos dois subpolígonos, excepto para os dois vértices que definem a diagonal, que pertence a ambos. Então, $n_1 + n_2 = n + 2$. Por indução, qualquer triangulação de P_i tem $n_i - 2$ triângulos o que implica que a triangulação de P tem $(n_1 - 2) + (n_2 - 2) = n - 2$ triângulos. ■

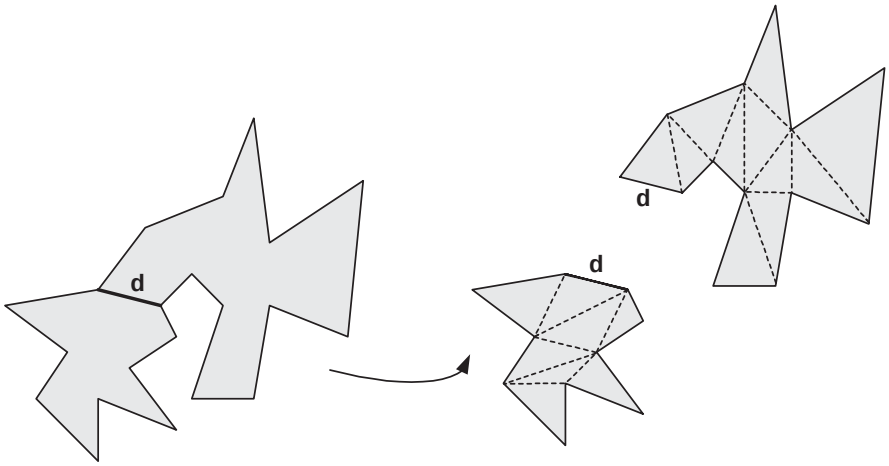


Figura 3.9: Ilustração da prova do Teorema 3.1.3.

Lema 3.1.3 *A soma dos ângulos internos de um polígono de n vértices é* $(n-2)\pi$.

Prova: Pelo teorema 3.1.4, existem $n-2$ triângulos numa triangulação de um polígono com n vértices. Como cada triângulo contribui com π para a soma dos ângulos, obtemos o pretendido. ■

Um importante conceito na teoria da triangulação é o de grafo dual da triangulação. O **dual** T de uma triangulação de um polígono P é obtido associando um nó no interior de cada triângulo de T e ligando dois nós se os seus triângulos correspondentes tiverem uma diagonal em comum (ver figura 3.10).

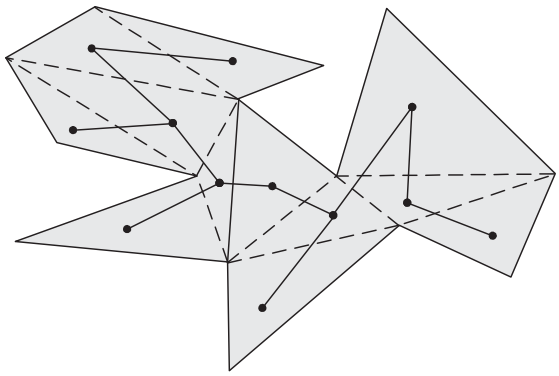


Figura 3.10: O grafo dual de uma triangulação.

Lema 3.1.4 *O dual T de uma triangulação é uma árvore*[3]*, onde cada nó tem no máximo grau três.*

Prova: Que cada nó tem no máximo grau três, é imediato pelo facto que cada triângulo tem no máximo três lados para partilhar.
Suponhamos que T não é uma árvore. Então deverá existir um ciclo, C, em T. Se este ciclo é desenhado como um caminho π no plano, ligando com segmentos de recta os pontos médios das diagonais partilhadas pelos triângulos cujos nós contêm C (para tornar o caminho específico), então deverá incluir alguns vértices do polígono, nomeadamente um ponto extremo de cada diagonal que é cruzado por π. Mas então π deverá incluir pontos exteriores ao polígono, pontos esses que estão na $fr(P)$. Isto contradiz a definição de polígono simples. ■

Os nós de grau 1 são as folhas de T; os nós de grau 2 estão situados nos caminhos da árvore; os nós de grau 3 são pontos de ramificação. Notemos que T é uma árvore binária[4] se a sua raiz for um nó de grau 1 ou 2.

Teorema 3.1.5 *Qualquer polígono simples com n vértices, $n > 3$, admite uma diagonal que divide o polígono em dois subpolígonos, nenhum dos quais com mais do que $\lceil 2n/3 \rceil + 1$ vértices.*

Prova: Dividindo recursivamente o polígono P por diagonais balanceadas[5] obtemos uma decomposição balanceada de P, que pode ser modelada por uma árvore de peso $O(log\, n)$. A existência de uma diagonal balanceada surge naturalmente uma vez que consideramos o grafo dual da triangulação. Este grafo dual é uma árvore com nós no máximo de grau três. As diagonais da triangulação correspondem às arestas da árvore dual, e assim, uma diagonal balanceada corresponde a uma aresta cuja remoção particiona a árvore em duas subárvores, cada qual com no máximo $\lceil 2n/3 \rceil + 1$ nós.

■

Do teorema 2.1.6 (capítulo 2), segue imediatamente o seguinte teorema:

[3]Uma árvore é um grafo conexo sem ciclos.

[4]Árvore em que cada vértice tem grau inferior ou igual a 2.

[5]Diagonais que particionam o polígonos em metades exactamente iguais.

Teorema 3.1.6 *Seja P um polígono com n vértices, $n > 3$, e T uma triangulação de P. Então pelo menos dois triângulos de T formam orelhas de P.*

Prova: A prova é feita por indução no número de vértices n de P. Se $n = 4$ então P é um quadrilátero e os dois triângulos de T são orelhas de P. Suponhamos que $n \geq 5$. Particionemos P em dois subpolígonos, P_1 e P_2 através de uma diagonal arbitrária d de T. Sejam T_1 e T_2 as triangulações de P_1 e P_2, respectivamente, obtidas através da restrição da triangulação T a P_1 e P_2. Pela hipótese de indução cada um dos subpolígonos P_1 e P_2 é um triângulo ou possuem duas orelhas formadas por triângulos em T_1 e T_2, respectivamente. Pelo menos um desses (ou possivelmente os dois triângulos de T_1) é uma orelha de P. Analogamente, pelo menos um desses (ou possivelmente os dois triângulos de T_2) é uma orelha de P. Como este triângulos são disjuntos, fica provado o pretendido. ■

A uma triangulação T de um polígono P, podemos associar um grafo $GT(P) = (V,\ E)$ da seguinte forma:

- o conjunto dos vértices V de $GT(P)$ é o conjunto dos vértices de P.
- Existe uma aresta em E unindo os vértices u e v de $GT(P)$ se o segmento $[uv]$ faz parte da triangulação de T.

Podemos definir uma triangulação T, de um conjunto de pontos S, do plano, como sendo o grafo máximo do plano tendo S como vértices.

3.1.2 Triangulação por cortes de orelhas

A triangulação por *cortes de orelhas* é um dos algoritmos que, apesar de ter complexidade $O(n^2)$, é um dos de mais simples implementação. Recordemos que uma orelha de um polígono, é um triângulo formado por três vértices consecutivos v_{i-1}, v_i e v_{i+1}, tal que o triângulo formado pertence totalmente ao interior do polígono. Já sabemos que o segmento que une v_{i-1} a v_{i+1} é uma diagonal. Ao vértice v_i chamamos **extremidade da orelha**. Pelo teorema 2.1.6 sabemos que um polígono com pelo menos quatro vértices têm pelo menos duas orelhas que não se sobrepõem. Este teorema sugere uma aproximação recursiva para a triangulação. Se conseguirmos localizar uma

orelha num polígono, com pelo menos quatro vértices, e removê-la obtemos um polígono com $n-1$ vértices, podendo o processo ser repetido. Uma implementação directa desta recursividade levar-nos-ia a um algoritmo de complexidade $O(n^3)$. Mas, estando atento aos pormenores, conseguimos melhorar a complexidade para $O(n^2)$.

- O primeiro passo é guardar o polígono como uma lista duplamente ligada, de maneira a que se consiga remover rapidamente as extremidades das orelhas. A construção desta lista tem complexidade linear.
- O segundo passo é percorrer os vértices e encontrar as orelhas. Para cada vértice v_i e o correspondente triângulo $\triangle(v_{i-1},\ v_i,\ v_{i+1})$, testar todos os outros vértices para verificar se há mais algum dentro do triângulo, isto é, se o triângulo pertence ao interior do polígono. Note-se que a indexação dos vértices é módulo n, pelo que $v_n \equiv v_0$ e $v_{-1} \equiv v_{n-1}$. Se não houver nenhum vértice dentro do triângulo, então temos uma orelha.

É suficiente considerarmos somente os vértices côncavos (reflexos) no teste do triângulo. A estrutura de dados para o polígono, mantém quatro listas duplamente ligadas simultaneamente, usando um vector em vez de estruturas dinâmicas (por exemplo, ponteiros). Os vértices do polígono são guardados numa lista cíclica, os vértices convexos são guardados numa lista linear, tal como os vértices côncavos e as extremidades das orelhas que são guardadas numa lista cíclica.

Uma vez construídas as listas iniciais para os vértices reflexos e para as extremidades das orelhas, estas serão removidas uma de cada vez. Se v_i for uma extremidade removida, então a configuração da aresta dos vértices adjacentes v_{i-1} e v_{i+1} pode mudar. Se um vértice adjacente for convexo, ele continuará convexo. Se um vértice adjacente for uma extremidade de uma orelha, não continuará necessariamente a ser uma extremidade após a remoção de v_i. Se o vértice adjacente for côncavo, é possível que se torne convexo e, possivelmente, uma extremidade de uma orelha. Assim, após a remoção de v_i, se um vértice adjacente é convexo devemos testar se é uma extremidade, percorrendo os vértices côncavos e testando se esse vértice pertence ao triângulo. Há $O(n)$ orelhas[6]. Cada actualização de um vértice adjacente envolve um novo teste. Este processo tem complexidade $O(n)$ por cada actualização. Assim, o processo total de remoção tem complexidade $O(n^2)$.

[6]Isto significa que o número de orelhas é proporcional ao número de vértices.

O exemplo seguinte, mostra como o algoritmo está estruturado. A lista inicial dos vértices convexos é $C = \{v_0, v_1, v_3, v_4, v_6, v_9\}$, a lista inicial de vértices reflexos é $R = \{v_2, v_5, v_7, v_8\}$ e a lista inicial da extremidade das orelhas é $E = \{v_3, v_4, v_6, v_9\}$. A figura 3.11 mostra o polígono simples considerado.

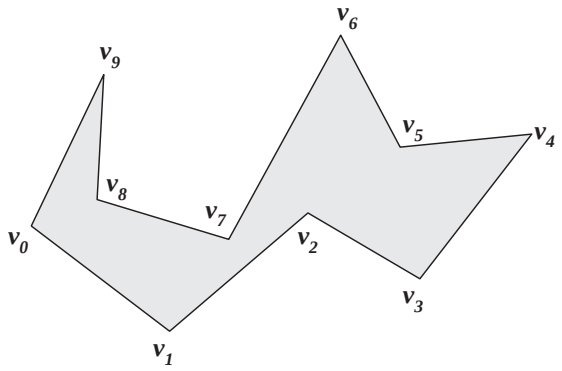


Figura 3.11: Polígono simples que irá ser triangulado pelo método da *Triangulação por Cortes de Orelhas*.

A extremidade da orelha de v_3 é removida, então o primeiro triângulo na triangulação é $T_0 = (v_2, v_3, v_4)$. A figura 3.12 mostra o novo polígono com a nova aresta (a negrito).

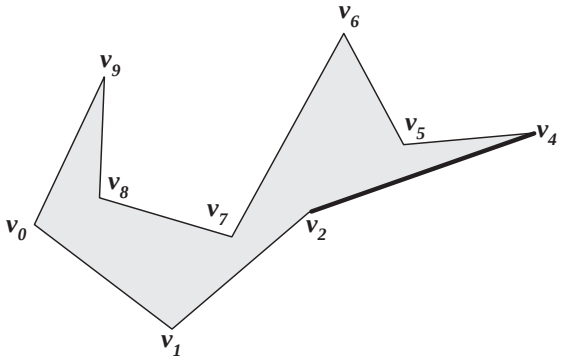


Figura 3.12: A orelha com extremidade v_3 foi removida.

O vértice adjacente v_2 era reflexo e continua a ser. O vértice adjacente v_4 era uma extremidade de uma orelha e continua a ser. A lista R, dos vértices reflexos, continua igual, mas a lista das extremidade das orelhas é agora $E = \{v_4, v_6, v_9\}$ (foi removido

v_3). A orelha seguinte a ser removida será o triângulo $T_1 = (v_2, v_4, v_5)$. A figura 3.13 (a) mostra o novo polígono com a nova aresta (a negrito).

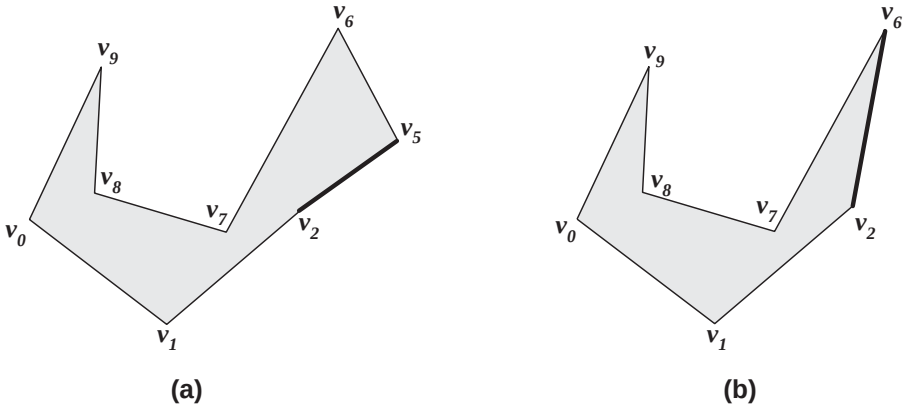


Figura 3.13: (a) A orelha com extremidade v_4 foi removida.
(b) A orelha com extremidade v_5 foi removida.

O vértice adjacente v_2 era reflexo e continua a ser. O vértice adjacente v_5 era reflexo, mas agora é convexo. Testa-se, então, se é extremidade de uma orelha. O vértice v_5 é removido da lista $R = \{v_2, v_7, v_8\}$ e é adicionado à lista E, das extremidades das orelhas, $E = \{v_5, v_6, v_9\}$ (adicionou-se v_4, removeu-se v_5).

A orelha com extremidade no vértice v_5 é removida. O próximo triângulo é $T_2 = (v_2, v_5, v_6)$. A figura 3.13 (b) mostra o novo polígono com a nova aresta (a negrito).

O vértice adjacente v_2 era reflexo mas, neste momento, é convexo. O vértice v_7 está dentro do triângulo (v_1, v_2, v_6), como tal, o vértice v_2 não é uma orelha. O vértice adjacente v_6 era uma orelha e assim permaneceu. A nova lista de vértices reflexos é $R = \{v_7, v_8\}$ (foi v_2 e a nova lista de extremidades de orelhas é $E = \{v_6, v_9\}$ (removido v_5).

O vértice v_6, que é extremidade de uma orelha, foi removido. O próximo triângulo é $T_3 = (v_2, v_6, v_7)$. A figura 3.14 (a) mostra o novo polígono com a nova aresta (a negrito).

O vértice adjacente v_2 era convexo e continuou convexo. Não era extremidade de uma orelha mas passou a sê-lo. O vértice v_7 permaneceu reflexo. A lista R não sofreu

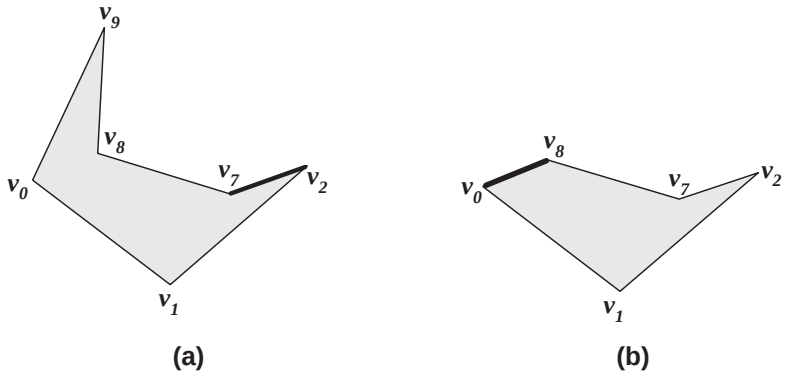


Figura 3.14: (a) A orelha com extremidade v_6 foi removida.
(b) A orelha com extremidade v_9 foi removida.

alterações, mas a lista E é agora $E = \{v_9, v_2\}$ (adicionou-se v_2, removeu-se v_6). Os elementos da lista E aparecem por esta ordem porque a extremidade da nova orelha foi adicionada antes da anterior ter sido removida. Antes da remoção da orelha "antiga"ela era considerada a primeira da lista.

A extremidade v_9 foi removida. O próximo triângulo da triangulação é $T_4 = (v_8, v_9, v_0)$. A figura 3.14 (b) mostra o novo polígono com a nova aresta (a negrito).

O vértice adjacente v_8 era reflexo, mas passou a ser convexo e também uma extremidade de uma orelha. O vértice adjacente v_0 era convexo e assim permaneceu, além disso não era extremidade de uma orelha mas tornou-se numa. A nova lista de vértices reflexos é $R = \{v_7\}$ e a nova lista de extremidades de orelhas é $E = \{v_0, v_2, v_8\}$ (adicionou-se v_8, v_0 e removeu-se v_9).

A extremidade da orelha, v_0 foi removida. O próximo triângulo da triangulação é $T_5 = (v_8, v_0, v_1)$. A figura 3.15 (a) mostra o novo polígono com a nova aresta (a negrito).

Ambos os vértices adjacentes, v_8 e v_1 eram convexos e continuaram a ser. Mas o vértice v_8 permaneceu como uma extremidade de uma orelha ao contrário do vértice v_1. A lista de vértices reflexos não se alterou. À lista das extremidades das orelhas, foi removido o vértice v_0, $E = \{v_2, v_8\}$.

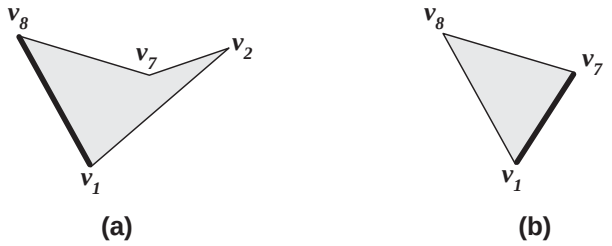


Figura 3.15: (a) A orelha com extremidade v_0 foi removida.
(b) A orelha com extremidade v_2 foi removida.

Finalmente, a extremidade da orelha, v_2 foi removida. O próximo triângulo na triangulação é $T_6 = (v_1, v_2, v_7)$. A figura 3.15 (b) mostra o novo polígono com a nova aresta (a negrito).

Por esta altura não é necessário actualizar quer a lista dos vértices reflexos, quer a lista das extremidades das orelhas, pois detecta-se que apenas restam três vértices. O último triângulo na triangulação é $T_7 = (v_7, v_8, v_1)$. A triangulação completa é mostrada na figura 3.16.

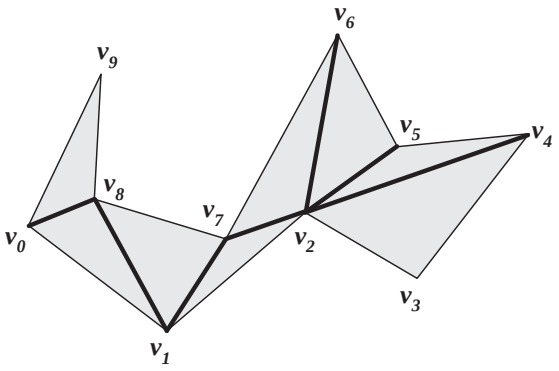


Figura 3.16: A triangulação completa do polígono simples.

3.1.3 O algoritmo de triangulação de Lennes

Este algoritmo baseia-se na técnica de inserção recursiva de diagonais. Nele, procuramos um vértice v_i que defina com os seus vértices adjacentes, v_{i-1} e v_{i+1}, um

ângulo convexo e, a partir dele, avaliamos se:

- o triângulo formado por estes pontos não contém outros vértices do polígono no seu interior, então v_{i-1} e v_{i+1} definem uma **diagonal**;
- existe pelo menos um vértice dentro do triângulo, então o segmento que liga v_i ao vértice mais próximo dele define uma **diagonal da triangulação**.

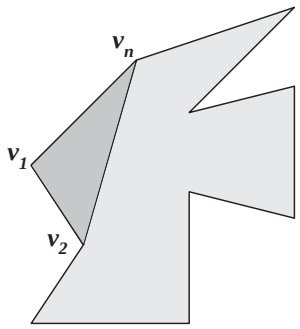


Figura 3.17: $[v_2v_n]$ é uma diagonal da triangulação.

Após esta verificação, o polígono foi dividido em dois outros (um triângulo e um polígono de $(n-2)$ lados ou dois polígonos de lados maiores ou iguais a 3). Portanto, devemos aplicar o algoritmo recursivamente a cada polígono até obter somente triângulos.

O algoritmo de Lennes:

Entrada: um polígono $P = (v_0, ..., v_{n-1})$.

Saída: uma triangulação de P por inserção recursiva de diagonais.

1. Se P for triângulo, pare.
2. Senão, determinar um vértice de P tal que o ângulo correspondente seja convexo. Suponhamos, sem perda de generalidade, que v_1 é um vértice com esta propriedade.

3. Se o conjunto V, dos vértices de P que estão contidos no triângulo $\triangle$ (v_1,v_2,v_n) é vazio então:

4. Faça $V_1 = v_1v_2v_n$ e $V_2 = v_2v_3...v_n$. Senão,

5. Determine entre os vértices em V o vértice v_k cuja distância a v_1 no triângulo $\triangle$ (v_1,v_2,v_n) é mínima.

6. Faça $V_1 = v_1v_2...v_n$ e $V_2 = v_1v_kv_{k+1}...v_n$

7. Aplique, recursivamente, o algoritmo a V_1 e V_2.

Cada execução do passo 3 tem tempo $O(n)$ e gera uma das $(n-3)$ diagonais da triangulação. Logo, o algoritmo tem complexidade $O(n^2)$.

3.1.4 O algoritmo de triangulação de Seidel

Este é um algoritmo aleatório que faz a triangulação a partir da decomposição sucessiva do polígono simples original, P, em polígonos cada vez mais simples até encontrar triângulos.

Primeiro, o algoritmo decompõe o polígono em trapézios. Isto é feito tomando as arestas de P por uma ordem aleatória. Esses segmentos são adicionados um a um e incrementalmente constroem os trapézios da seguinte forma: por cada aresta de P traçam-se segmentos horizontais que partem de cada vértice e terminam quando intersectam uma aresta. Note-se que, podemos obter triângulos ao invés de trapézios como mostra a figura 3.18. Depois, decompomos esses trapézios em polígonos y-monótonos. Encontramos esses polígonos verificando se dois vértices do polígono original estão do mesmo lado do trapézio. Finalmente decompomos os polígonos y-monótonos em triângulos, traçando continuamente as diagonais do polígono y-monótono.
Resumindo, o algoritmo é o seguinte:

1. Decomponha o polígono em trapézios.

2. Decomponha os trapézios em polígonos y-monótonos.

3. Faça a triangulação dos polígonos y-monótonos.

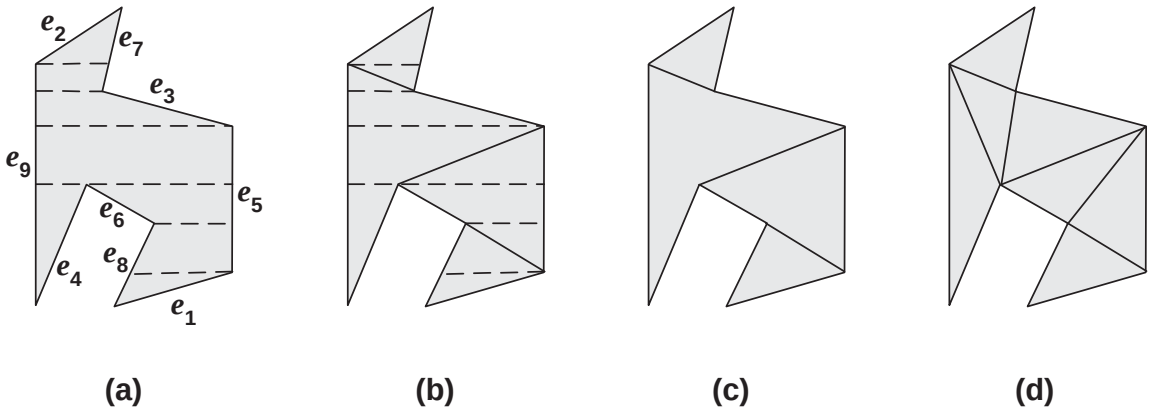


Figura 3.18: (a) P particionado em trapézios. As arestas de P foram escolhidas de forma aleatória. (b) Partição de P em polígonos y-monótonos. (c) P particionado em polígonos y-monótonos. (d) Triangulação dos polígonos y-monótonos.

No passo 1, o número de trapézios é linear ao número de segmentos. Seidel provou [95] que cada permutação de $e_1,\ e_2, \ldots, e_n$ é praticamente igual, logo a construção dos trapézios tem complexidade $O\,(n\,log^*n)$. Nos passos 2 e 3 ambas as operações podem ser feitas em $O\,(n)$. Logo o tempo total do algoritmo é $O\,(n\,log^*n)$.

3.2 Partição de um polígono em partes monótonas

O próximo algoritmo, particiona um polígono dado em peças (polígonos) monótonas usando um algoritmo que se deve a Lee e Preparata [65]. Este algoritmo recorre à utilização da técnica da linha de varrimento. Descreveremos como, utilizando este método, podemos obter um algoritmo de complexidade de tempo $O(n\,log\,n)$ e de espaço $O(n)$ para triangular um polígono.

Consideremos um polígono simples, P, com n vértices. O teorema 3.1.4 afirma que existe sempre uma triangulação de P. Podemos fazer a prova deste teorema de uma forma construtiva o que nos levará a um algoritmo de triangulação recursivo: encontrar uma diagonal e triangular os dois subpolígonos resultantes recursivamente. Para encontrar uma diagonal, consideramos o vértice mais à esquerda de P, v, e

tentamos ligar os seus vizinhos u e w; se não conseguirmos, ligamos v ao vértice mais afastado do segmento $[uw]$, pertencente ao interior do triângulo definido por u, v e w. Este procedimento de encontrar uma diagonal leva um tempo linear. Esta diagonal pode decompor P num triângulo e num polígono com $n-1$ vértices. Na verdade, se formos bem sucedidos na ligação de u com w, ou seja se o segmento $[uw]$ for uma diagonal, teremos a divisão num triângulo e num polígono com $n-1$ vértices. Como consequência, o algoritmo de triangulação, no pior caso, levará um tempo quadrático. Poder-se-á melhorá-lo? Para algumas classes de polígonos, sim. Por exemplo, triangular polígonos convexos é mais fácil: basta considerar um vértice do polígono e construir diagonais desde esse vértice até todos os outros, exceptuando os vizinhos. Este procedimento leva um tempo linear. Então, uma possível estratégia para triangular um polígono que não seja convexo, seria primeiro decompô-lo em partes convexas e depois triangular essas partes. Infelizmente é algoritmicamente complicado particionar um polígono em partes convexas. Torna-se mais fácil particioná-lo em partes monótonas.

Recordemos que um polígono que seja monótono relativamente ao eixo das ordenadas chama-se y-monótono. Uma propriedade característica desta classe de polígonos é a seguinte: se percorrermos o polígono desde o vértice com maior ordenada até ao vértice de menor ordenada ao longo da fronteira da cadeia poligonal esquerda (ou da cadeia poligonal direita), então mover-nos-emos sempre para baixo ou na horizontal, nunca para cima.

Para decompor um polígono em partes monótonas podemos proceder da seguinte maneira: imaginemos que começamos a percorrer o polígono iniciando no vértice com maior ordenada até ao vértice de ordenada menor, quer seja pela fronteira da cadeia poligonal esquerda ou da cadeia poligonal direita. Um vértice que esteja na direcção que estamos a percorrer mude de cima para baixo, ou de baixo para cima, chama-se **vértice de viragem**. Para particionarmos P (em partes y-monótonas) deveremos eliminar esses vértices de viragem. Isto pode ser feito, adicionando diagonais.

- Se num vértice de viragem v, as arestas que lhe incidem se direccionam para baixo dele e o interior do polígono está acima de v, então devemos escolher uma diagonal que parta de v e se direccione para cima. Esta diagonal divide o polígono em dois. O vértice v irá aparecer em ambas as partes. Além disso, em ambas as partes, v tem uma aresta que se direcciona para baixo (aresta essa que estava

originalmente em P) e outra que se direcciona para cima (a diagonal). Assim, v deixa de ser um vértice de viragem em ambas as partes (ver figura 3.19).

- Se num vértice de viragem v, as arestas que lhe incidem se direccionam para cima dele e o interior do polígono está abaixo de v, devemos escolher uma diagonal que parta de v e se direccione para baixo.

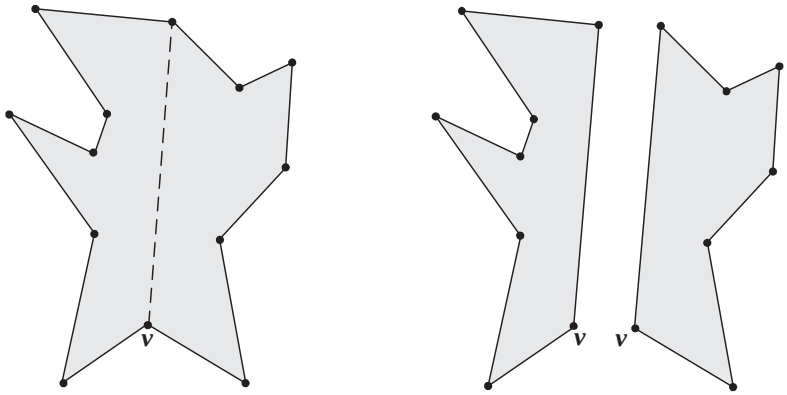


Figura 3.19: O vértice de viragem v, tem as arestas incidentes direccionadas para baixo. É traçada uma diagonal para cima, que divide o polígono em dois. O vértice v deixou de ser de viragem.

Aparentemente existem diferentes tipos de vértices de viragem. Se quisermos definir cuidadosamente esses diferentes tipos, devemos prestar especial atenção aos vértices com a mesma ordenada. Como tal, definamos as noções de "acima"e "abaixo"da seguinte maneira: um ponto p está abaixo de um outro ponto q se $p_y < q_y$ ou $p_y = q_y$ e $p_x > q_x$; e p está acima de q se $p_y > q_y$ ou $p_y = q_y$ e $p_x < q_x$.

Podemos distinguir cinco tipos de vértices num polígono P (ver figura 3.20). Quatro desses vértices são de viragem: os vértices de partida, os vértices de quebra, os vértices finais e os vértices de união. Definem-se da seguinte maneira:

- Um **vértice**, v, diz-se de **partida** se os seus dois vértices vizinhos estão abaixo de v e o ângulo interior em v é menor que π.

– Se o ângulo for maior que π o **vértice** diz-se de **quebra** (se ambos os vizinhos estão ambos abaixo de v então o ângulo nunca pode ser π).

- Um **vértice**, w, diz-se **final** se os seus dois vértices vizinhos estão acima de v e o ângulo interior em v é menor que π.

  – Se o ângulo for maior que π o **vértice** diz-se de **união**.

Caso os vértices não sejam de viragem dizem-se que são regulares. Assim, um **vértice** é **regular** se tiver um vértice vizinho acima dele e o outro vértice vizinho abaixo dele.

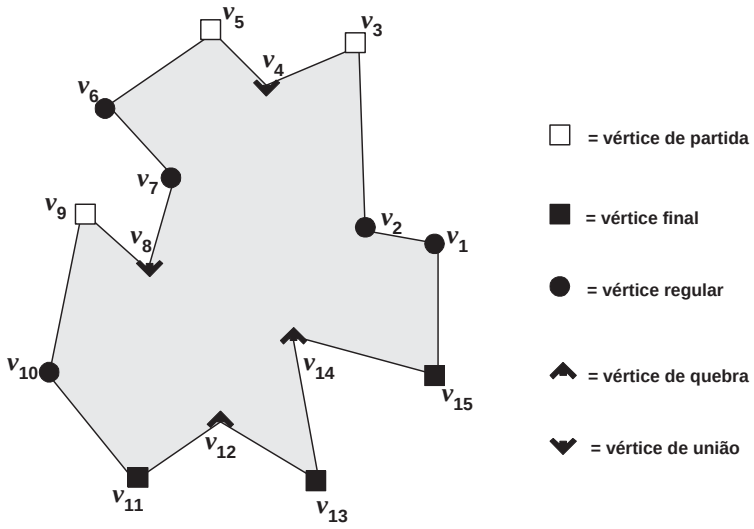


Figura 3.20: Cinco tipos de vértices.

As designações para os vértices foram assim escolhidas pois o algoritmo de partição em partes monótonas usa um plano de varrimento descendente, mantendo sempre a intersecção da linha de varrimento com o polígono. Quando esta linha atinge um vértice de quebra, uma componente da intersecção quebra, quando atinge um vértice de união, duas componentes unem-se e por aí adiante.

Lema 3.2.1 *Um polígono é y-monótono se ele não tem vértices de quebra nem vértices de união.*

Prova: Suponhamos que P é um polígono não y-monótono. Pretendemos mostrar que P contém um vértice de quebra ou de união.
Como P é não y-monótono, existe uma recta horizontal, l, que intersecta P em mais do que uma componente conexa. Podemos escolher l tal que a componente mais à esquerda seja um segmento e não um simples ponto. Seja p o extremo esquerdo deste segmento e seja q o extremo direito. Começando em q, seguiremos a fronteira de P de tal forma que o polígono está sempre à nossa esquerda. Nalgum ponto, digamos r, a fronteira irá intersectar novamente l. Se $r \neq p$ como na figura 3.21 (a), então o vértice mais alto que encontramos enquanto estávamos a percorrer a fronteira de q até r, é necessariamente um vértice de quebra.

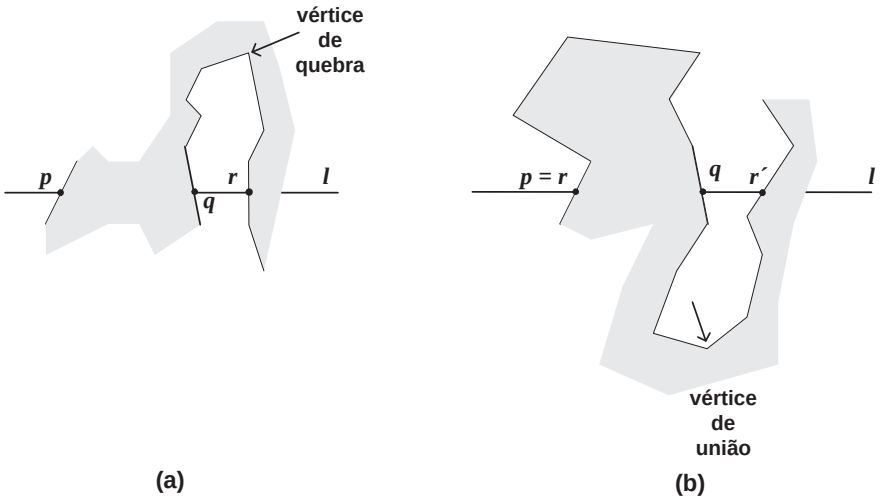


Figura 3.21: Dois casos da prova do lema 3.2.1.

Se $p = r$, como na figura 3.21 (b), percorremos novamente a fronteira de P a partir de q, mas desta vez noutro sentido. Como anteriormente, a fronteira irá intersectar l novamente. Seja r' o ponto onde esta situação ocorre. Não podemos ter $r' = p$, pois isto significa que a fronteira de P intersecta l apenas duas vezes, contrariando a hipótese de que l intersecta P em pelo menos duas componentes conexas. Assim, $r' \neq p$, implicando que o vértice mais baixo que encontramos no caminho de q até r' é necessariamente um vértice de união. ■

O lema 3.2.1 implica que um polígono P terá sido particionado em partes y-monótonas uma vez que eliminámos os seus vértices de quebra e de união. Para tal, vai-se adicionando diagonais que "vão para cima"a partir de vértices de quebra, e que "vão

para baixo"a partir de vértices de união. Obviamente que estas diagonais não se devem intersectar. Ficamos, assim, com o polígono particionado em partes y-monótonas.

Para adicionamos diagonais para os vértices de quebra usamos um método de varrimento do plano. Seja $v_1, v_2, ..., v_n$ uma enumeração no sentido contrário aos dos ponteiros do relógio, dos vértices de um polígono P. Sejam $e_1, e_2, ..., e_n$ as arestas de P, onde $e_i = [v_i v_{i+1}]$ para $1 \leq i < n$ e $e_n = [v_n v_1]$. O algoritmo de varrimento do plano move uma linha imaginária, l, designada por *linha de varrimento*, que varre o objecto ou objectos em análise segundo uma certa direcção e sentido, enquanto se vai recolhendo informação e efectuando algum tipo de processamento. Esta linha pára em certos pontos eventos[7]. Neste caso estes pontos serão os vértices de P; mais nenhum ponto evento será criado durante o varrimento. Os pontos eventos (vértices) são guardados numa lista[8] ordenada Q. Caso haja dois vértices com a mesma ordenada, então o vértice que tem menor abcissa (ou seja, o vértice mais à esquerda) tem prioridade. Desta maneira, os pontos eventos seguintes serão encontrados num tempo $O(log\, n)$. (Como os pontos eventos são somente os vértices de P podemos também ordená-los pelas suas ordenadas e depois usar essa lista ordenada para encontrar o próximo ponto evento num tempo $O(1)$.)

O objectivo do varrimento é adicionar diagonais desde cada vértice de quebra até um vértice que está acima dele. Suponhamos que a linha de varrimento, l, intersecta um vértice de quebra v_i. Qual deverá ser o vértice que deverá ligar-se a v_i? Um bom candidato é um vértice próximo de v_i, pois provavelmente podemos ligar v_i a esse vértice sem intersectarmos nenhuma aresta de P. Mais precisamente, seja e_j a aresta imediatamente à esquerda de v_i sobre a linha de varrimento, e seja e_k a aresta imediatamente à direita de v_i. Então podemos sempre ligar v_i ao vértice mais baixo entre e_j e e_k e acima de v_i. Se esse vértice não existir, então podemos ligar v_i ao vértice final mais alto de e_j ou ao vértice final mais alto de e_k. Chamamos a esse vértice o ajudante de e_j e denotamo-lo por *ajudante* (e_j). Definimos então **vértice ajudante** como o vértice mais baixo (isto é, de menor ordenada) acima da linha de varrimento indexvarrimento!linha de tal modo que o segmento horizontal que liga o vértice a e_j está no interior de P. Notemos que o *ajudante* (e_j) pode ser ele próprio o vértice extremo mais elevado.

[7]Pontos particulares onde é necessário fazer uma actualização no algoritmo.

[8]A lista Q é, portanto, uma lista ordenada que guarda todos os vértices de P, vértices esses que chamamos de pontos eventos.

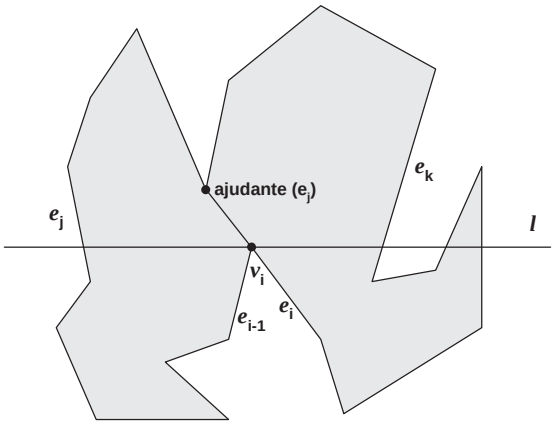


Figura 3.22: Exemplo de uma diagonal quando o vértice é de quebra.

Sabemos, agora, como eliminar os vértices de quebra: ligamo-los ao *ajudante* da aresta à sua esquerda. Já os vértices de união parecem ser mais difícil eliminá-los, porque precisam de uma diagonal até a um vértice que esteja mais abaixo do que eles. Até que a parte de P abaixo da linha de varrimento não tenha sido "varrida", não podemos adicionar uma diagonal quando encontramos um vértice de união. Felizmente este problema é mais fácil de resolver do que parece à primeira vista. Suponhamos que a linha de varrimento alcança um vértice de união, v_i. Sejam e_j e e_k as arestas imediatamente à direita e à esquerda do vértice v_i pertencente à linha de varrimento, respectivamente. Observemos que v_i torna-se o novo vértice ajudante de e_j quando a linha de varrimento o alcançar. Pretendemos ligar v_i ao vértice mais alto abaixo da linha de varrimento que está entre e_j e e_k. (Isto é exactamente o oposto ao que foi feito para vértices de quebra, onde ligámos o vértices mais baixo acima da linha de varrimento que estava entre e_j e e_k). O que é natural pois vértices de união são vértices de quebra mas virados para baixo. Claro que não sabemos qual é o vértice mais alto abaixo da linha de varrimento quando alcançamos v_i, mas é fácil encontrá-lo mais tarde pois, quando atingirmos o vértice v_m que substituiu v_i como ajudante de e_j, será este o vértice que procuramos. Então, sempre que substituímos algum vértice ajudante de alguma aresta, verificamos se o vértice ajudante antigo é vértice de união; se for, adicionamos uma diagonal entre o vértice ajudante antigo e o novo. Esta diagonal é sempre adicionada quando o novo vértice ajudante for um vértice de quebra, por forma a eliminarmos este vértice. Se o vértice ajudante velho for um vértice de união, eliminamos assim um vértice de quebra e um vértice de união com a mesma diagonal.

Poderá também suceder que o ajudante de e_j nunca seja substituído abaixo de v_i. Neste caso, ligamos v_i ao vértice mais baixo de e_j.

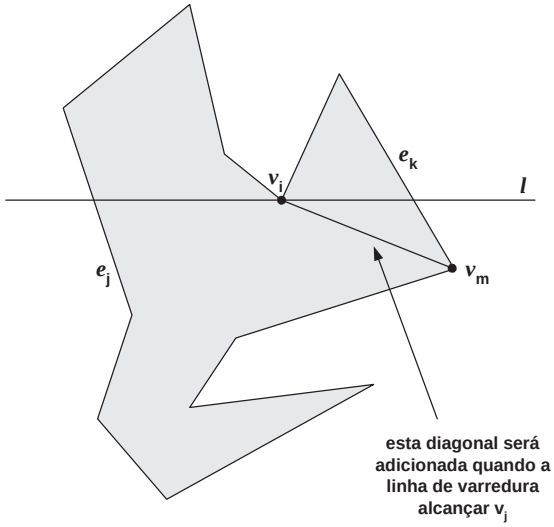


Figura 3.23: Exemplo de uma diagonal quando o vértice é de união.

Em seguida, pretende-se encontrar uma aresta à esquerda de cada vértice. Para tal, guardaremos as arestas de P que intersectam a linha de varrimento, nas folhas de uma árvore binária dinâmica de procura, Γ. A ordem da esquerda para a direita das folhas de Γ corresponde à ordem da esquerda para a direita das arestas. Como estamos só interessados nas arestas da esquerda dos vértices de quebra e de união, só precisamos de guardar em Γ arestas que têm o interior de P à sua direita. Com cada uma das arestas em Γ, guardaremos o seus ajudantes. A árvore Γ e os ajudantes, armazenados com as arestas, indicam-nos a fase que em estamos no algoritmo da linha de varrimento. Essa fase vai mudando à medida que a linha de varrimento se move: as arestas iniciam ou param de intersectar a linha de varrimento e o vértice ajudante de uma aresta pode ser substituído.

O algoritmo particiona P em dois subpolígonos que têm que ser tratados numa fase posterior. Para termos um acesso mais fácil a estes subpolígonos, devemos guardar a subdivisão induzida por P e as diagonais que foram adicionadas numa lista de arestas duplamente ligada, D. Assumimos que P é inicialmente uma lista de arestas duplamente ligada; se P for dado por uma forma diferente - por exemplo, por uma lista dos seus vértices ordenados no sentido anti-horário - construímos primeiro uma lista de

arestas duplamente ligada de P. As diagonais processadas para os vértices de quebra e de união são adicionadas à lista de arestas duplamente ligadas. Para aceder a esta lista, usamos ponteiros cruzados entre as arestas na estrutura actual e os correspondentes vértices na lista de arestas duplamente ligada. Adicionar uma diagonal pode ser feito em tempo constante com simples manipulações de ponteiros. O algoritmo é, então, o seguinte:

Algoritmo *FazMonotono(P)*

Entrada: Um polígono simples P guardado numa lista de arestas duplamente ligada, Q.
Saída: Uma partição de P em subpolígonos monótonos, guardada em D.

1. Construa uma lista ordenada (usando as suas ordenadas), Q, dos vértices de P. Se dois vértices tiverem a mesma ordenada, o que tem menor abcissa tem prioridade.
2. Inicialize uma árvore vazia binária de pesquisa, Γ.
3. **enquanto** Q não está vazia
4. **faz** remova de Q o vértice v_i com maior prioridade.
5. Chame o procedimento adequado para processar o vértice, dependendo do seu tipo.

Em seguida descrevemos como lidar com os pontos eventos. Há sempre duas coisas que devemos fazer quando lidamos com um vértice. Primeiro, devemos verificar se temos que adicionar uma diagonal. Este é sempre o caso para o vértice de quebra e também quando substituímos o vértice ajudante de uma aresta sendo o vértice ajudante anterior um vértice de união . Segundo, devemos actualizar a informação na árvore actual, Γ. Os algoritmos para cada tipo de vértice evento são dados a seguir. Podemos usar o exemplo da figura 3.24 para perceber o que se passa nos diferentes casos.

Procedimento *SeguraVerticeFinal*(v_i)

1. **se** *ajudante*(e_{i-1}) é um vértice de união
2. **então** insira uma diagonal que liga v_i ao *ajudante*(e_{i-1}) em D.

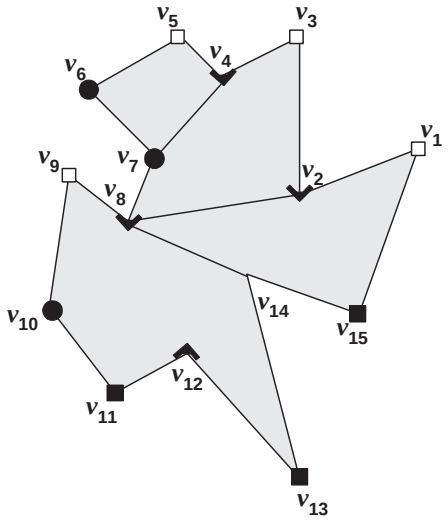


Figura 3.24: Exemplo de aplicação dos algoritmos para os diferentes tipos de vértices.

3. Apague e_{i-1} de Γ.

No exemplo da figura 3.24, quando alcançamos o vértice v_{15}, o *ajudante* da aresta e_{14} é v_{14}. Não é um vértice de união, portanto não precisamos de inserir uma diagonal.

Procedimento *SeguraVerticeQuebra*(v_i)

1. Procure em Γ encontrar a aresta e_j directamente à esquerda de v_i.
2. Insira a diagonal que liga v_i ao *ajudante*(e_j) em D.
3. *ajudante*(e_j) $\longleftarrow$ v_i
4. Insira e_i em Γ e defina o *ajudante*(e_i) para v_i.

Para o vértice de quebra v_{14} no exemplo da figura 3.24, e_9 é a aresta à esquerda. O seu *ajudante* é v_8, então adicionamos uma diagonal desde v_{14} até v_8.

Procedimento *SeguraVerticeUniao*(v_i)

1. **se** *ajudante*(e_{i-1}) é um vértice de união
2. **então** insira uma diagonal que liga v_i ao *ajudante*(e_{i-1}) em D.
3. Apague e_{i-1} de Γ.
4. Procure em Γ encontrar a aresta e_j que está directamente à esquerda de v_i.
5. **se** *ajudante*(e_j) é um vértice de união
6. **então** insira a diagonal que liga v_i ao *ajudante*(e_j) em D.
7. *ajudante*(e_j) $\longleftarrow$ v_i

Para o vértice de união v_8 no exemplo da figura 3.24, o *ajudante* v_2 da aresta e_7 é um vértice de união, então adicionamos uma diagonal desde v_8 até v_2.

A única rotina que nos resta descrever é aquela que nos dá um vértice regular. O procedimento que devemos ter num vértice regular depende se P está à esquerda ou à direita do vértice.

Procedimento *SeguraVerticeRegular*(v_i)

1. **se** o interior de P está à direita de v_i
2. **então se** *ajudante*(e_{i-1}) é um vértice de união
3. **então** insira uma diagonal que liga v_i ao *ajudante*(e_{i-1}) em D.
4. Apague e_{i-1} de Γ.
5. Insira e_i em Γ e defina o *ajudante*(e_i) para v_i.
6. **senão** procure encontrar em Γ a aresta e_j directamente à esquerda de v_i.
7. **se** *ajudante*(e_j) é um vértice de união
8. **então** insira uma diagonal que liga v_i ao *ajudante*(e_j) em D.
9. *ajudante*(e_j) $\longleftarrow$ v_i

No exemplo da figura 3.24, no vértice regular v_6 adicionámos uma diagonal deste v_6 até v_4.

Falta apenas provar que o algoritmo *FazMonotono* faz correctamente as partições monótonas nos polígonos.

Lema 3.2.2 *O algoritmo* FazMonotono *adiciona diagonais que particionam P em subpolígonos monótonos.*

Prova: É fácil perceber que as partes em que P está particionado, não contêm vértices de quebra ou de união. Assim, pelo lema 3.2.1 são monótonas. Falta apenas provar que os segmentos adicionados são diagonais, ou seja, que não intersectam as arestas de P nem se intersectam entre si. Para isso, iremos mostrar que quando um segmento é adicionado, ele não intersecta nenhuma aresta de P nem intersecta nenhuma diagonal que já tivesse sido adicionada. Vamos provar este facto para o segmento adicionado no procedimento *SeguraVerticeQuebra*. A prova para os outros procedimentos (*SeguraVerticeFinal*, *SeguraVerticeRegular* e *SeguraVerticeUniao*) é semelhante. Assume-se que não há dois vértices com a mesma ordenada; a extensão ao caso geral é imediata.

Consideremos o segmento $[v_m v_i]$ que é adicionado pelo algoritmo do procedimento *SeguraVerticeQuebra* quando v_i é alcançado. Seja e_j a aresta à esquerda de v_i, e seja v_k a aresta à direita de v_i. Assim, o $ajudante(e_j) = v_m$ quando alcançamos v_i.

Podemos argumentar que $[v_m v_i]$ não intersecta qualquer aresta de P. Para mostrarmos este facto, consideremos o quadrilátero Q limitado por dois segmentos horizontais (que passam por v_m e v_i) e por e_j e e_k. Não existe nenhum vértice de P no interior de Q, caso contrário, v_m não poderia ser um *ajudante* de e_j. Suponhamos agora que havia uma aresta de P que intersectava $[v_m v_i]$. Nesse caso, a aresta não pode ter um ponto extremo em Q e as arestas do polígono não se intersectam; teria que intersectar o segmento horizontal que liga v_m a e_j ou o segmento horizontal que liga v_i a e_j. Ambos os casos são impossíveis, pois para os vértices v_m e v_i, a aresta e_j está imediatamente à esquerda. Assim, nenhuma aresta de P pode intersectar $[v_m v_i]$.

Consideremos agora que já tínhamos adicionado uma diagonal. Então não existem vértices de P no interior de Q e qualquer diagonal adicionada anteriormente deveria ter ambos os seus pontos extremos acima de v_i, logo não intersecta $[v_m v_i]$. ■

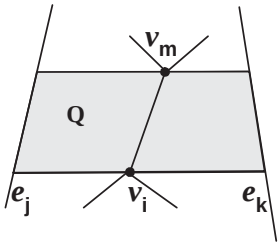


Figura 3.25: Ilustração da prova do lema 3.2.2.

Análise do algoritmo

Analisaremos agora o tempo de execução do algoritmo. Construir uma lista ordenada Q leva tempo $O(n)$ e inicializar Γ um tempo $O(1)$. Para processar um ponto evento durante o varrimento, fizemos uma operação em Q, no máximo uma ordenação, uma inserção, uma eliminação em Γ e inserimos no máximo duas diagonais em D. Listas ordenadas e árvores de pesquisa balanceadas permitem ordenar e actualizar num tempo $O(log\, n)$ e a inserção de uma diagonal em D leva tempo $O(1)$. Assim, obter um ponto evento leva tempo $O(log\, n)$ e o algoritmo é executado na totalidade em $O(n\, log\, n)$. O espaço utilizado pelo algoritmo é linear pois cada vértice é guardado no máximo uma vez em Q e cada aresta uma vez em Γ.

Assim, podemos enunciar o seguinte teorema:

Teorema 3.2.1 *Um polígono com n vértices pode ser particionado em polígonos y-monótonos em tempo $O(n\, log\, n)$ por um algoritmo que usa espaço $O(n)$.*

3.2.1 Triangulação de polígonos monótonos

Acabámos de verificar como particionar um polígono simples em partes y-monótonas num tempo $O(n\, log\, n)$. Nesta secção mostramos como os polígonos monótonos podem ser triangulados num tempo linear, através do algoritmo descrito por Garey et al [38]. Combinados, estes dois resultados implicam que qualquer polígono simples pode ser triangulado num tempo $O(n\, log\, n)$, que é um melhoramento relativamente ao algoritmo de tempo quadrático de Lennes, descrito na subsecção 3.1.3.

Seja P um polígono y-monótono com n vértices. Assumimos que P é estritamente y-monótono[9]. Assim, iremos sempre para baixo quando percorremos a cadeia poligonal esquerda ou direita de P, desde o vértice de maior ordenada até ao de menor ordenada. A propriedade que faz a triangulação de um polígono monótono simples é a seguinte: podemos trabalhar em P desde cima até baixo em ambas as cadeias, adicionando diagonais sempre que seja possível. Em seguida descrevemos os detalhes deste algoritmo.

O algoritmo trabalha com os vértices por ordem decrescente das suas ordenadas. Um facto fundamental do ponto de vista algorítmico envolvendo polígonos monótonos, é que os seus vértices podem ser ordenados em relação às suas ordenadas num tempo linear. Se dois vértices têm a mesma ordenada, então o que tem a menor abcissa terá prioridade. O algoritmo requer a utilização de uma pilha S como estrutura de dados auxiliar. A pilha inicialmente está vazia; mais tarde conterá os vértices de P que foram encontrados, mas poderá conter ainda mais diagonais. Quando processamos um vértice adicionamos o maior número possível de diagonais deste vértice até aos vértices que se encontram na pilha. Estas diagonais dividem P em triângulos. Os vértices que foram processados mas não separados - vértices que estão na pilha - estão na fronteira de uma parte de P que ainda precisa de ser triangulada. Destes vértices, o de menor ordenada, que é o que foi encontrado em último lugar, está no topo da lista, o segundo de menor ordenada é o segundo da pilha, e assim por diante. A parte do polígono que ainda precisa de ser triangulada, e que está abaixo do último vértice que foi encontrado até ao momento, tem uma forma particular: parece um funil virado ao contrário. Uma das fronteiras do funil é uma parte de uma aresta de P, e a outra fronteira é uma cadeia constituída por vértices reflexos, ou seja, os ângulos interiores são todos inferiores a π. Somente o vértice de maior ordenada, que está em baixo na pilha, é convexo. Esta propriedade permanecerá verdadeira até processarmos o vértice seguinte. Logo este procedimento trata-se de uma invariante do algoritmo.

Vejamos, agora, quais serão as diagonais que podemos adicionar quando processamos o próximo vértice. O próximo vértice a ser processado será v_j. Temos que distinguir dois casos: ou v_j está na mesma cadeia que os vértices reflexos que estão na pilha ou está na cadeia oposta.

- Se v_j estiver na mesma cadeia que os vértices reflexos que estão na pilha, desta fez

[9]Não existem vértices com a mesma ordenada.

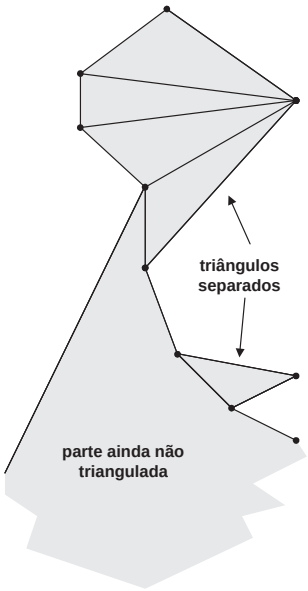


Figura 3.26: Parte do polígono que ainda precisa de ser triangulada, com aparência de um funil.

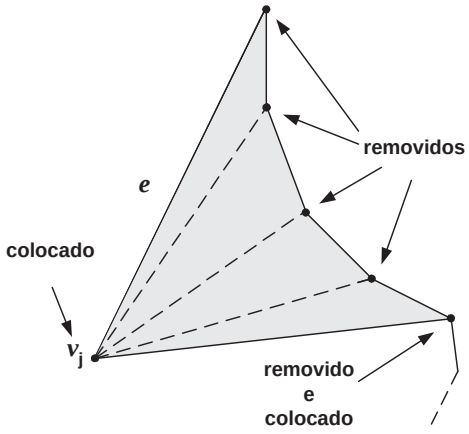


Figura 3.27: Triangulação da parte não triangulada.

não podemos adicionar diagonais desde v_j até todos os vértices da pilha. Apesar disso, aqueles vértices aos quais podemos ligar v_j estão todos consecutivos e estão

no topo da pilha, por isso procedemos da seguinte maneira: primeiro, tiramos um vértice da pilha; este vértice já está ligado a v_j por uma aresta de P. Depois, tiramos vértices da pilha e ligamo-los a v_j até encontrarmos um que não seja possível ligar. A verificação da diagonal que pode ser adicionada desde o vértice v_j até ao vértice v_k na pilha, pode ser feito olhando para v_j, v_k e o vértice precedente que foi tirado da pilha. Quando encontrarmos um vértice ao qual não podemos ligar v_j, repomos o vértice que tínhamos retirado anteriormente, na pilha. Este é o último vértice ao qual a diagonal é adicionada ou, se nenhuma diagonal foi adicionada, o vértice é vizinho de v_j na fronteira de P. Ver figura 3.28.

- Se v_j está na cadeia oposta então é o ponto extremo de menor ordenada da aresta, e, que limita o funil. Devido à forma do funil, podemos adicionar diagonais desde v_j até todos os vértices que estão na correntemente na pilha, excepto o último, aquele que está no fim da pilha. Este último vértice é o que tem maior ordenada da aresta e, portanto já se encontra ligado a v_j. Todos estes vértices saem da pilha. A parte não triangulada do polígono abaixo de v_j está limitada pela diagonal que liga v_j ao vértice previamente no topo da pilha e pela aresta de P que se extende para baixo a partir deste vértice, por isso, parece um funil e a invariante é preservada. Este vértice e v_j pertencem à parte que ainda não foi triangulada de P, portanto irão para a pilha.

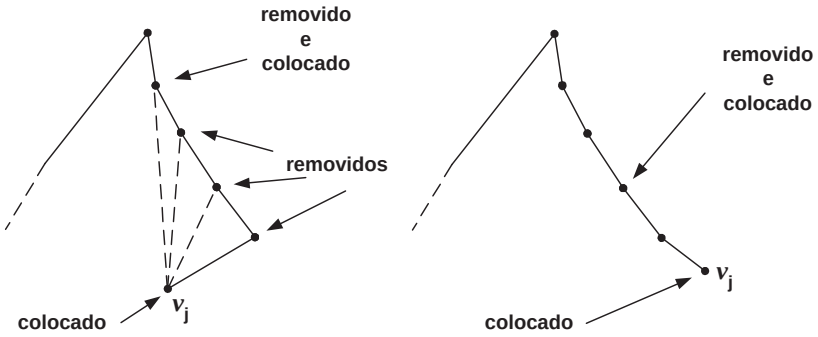


Figura 3.28: Dois casos em que o vértice adjacente está no mesmo lado que os vértices da cadeia reflexa que estão na pilha.

Depois disto, voltamos a colocar v_j na pilha. Em ambos os casos a invariante é reposta: um lado do funil é limitado por uma parte de uma aresta de P e o outro lado é limitado por uma cadeia de vértices reflexos.

Temos assim o seguinte algoritmo:

Algoritmo *TriangulaPoligonoMonotono*(P)

Entrada: um polígono P y-monótono guardado numa lista de arestas duplamente ligada D.

Saída: uma triangulação de P guardado numa lista de arestas duplamente ligada D.

1. Una os vértices da cadeia esquerda e os da cadeia direita de P numa sequência, ordenados por ordem decrescente das suas ordenadas. Se dois vértices tiverem a mesma ordenada, então o de menor abcissa tem prioridade. Seja $u_1, ..., u_n$ a sequência ordenada.
2. Inicialize um pilha vazia S, e adicione u_1 e u_2 à pilha S.
3. **for** $j \longleftarrow 3$ **até** $n-1$
4. **faz se** u_j e o vértice no topo de S estão em cadeias diferentes
5. **então** retire todos os vértices de S.
6. Insira em D a diagonal desde u_j até cada um dos vértices que foram tirados de S, excepto o último.
7. Coloque u_{j-1} e u_j em S.
8. **senão** Retire um vértice de S.
9. Tire os outros vértices de S assim como as diagonais que partem de u_j até aos vértices que estão no interior de P. Insira estas diagonais em D. Reponha o último vértice que foi tirado para S.
10. Coloque u_j em S.

11. Adicione diagonais desde u_n até todos os vértices da pilha excepto o primeiro e o último.

Análise do algoritmo

Analisemos o tempo que este algoritmo gasta para triangular um polígono y-monótono.

- O 1º passo tem tempo linear e o 2º um tempo constante.
- O ciclo **for** é executado $n-3$ vezes e uma execução pode gastar tempo linear, o que sugere que a complexidade do algoritmo é $O(n^2)$.
- Mas cada execução do ciclo **for** tem pelo menos dois vértices que são colocados. Assim, o número total de colocações incluindo as duas no passo 2, é limitado por $2n-4$.
- Como o número de eliminações não pode exceder o número de colocações o tempo total para todas as execuções do ciclo **for** é $O(n)$.
- O último passo do algoritmo tem também um tempo no máximo linear, portanto a complexidade do algoritmo é $O(n)$.

Podemos enunciar, assim, o seguinte teorema:

Teorema 3.2.2 *Um polígono estritamente y-monótono com n vértices pode ser triangulado em $O(n)$.*

3.2.2 Triangulação de um polígono em $O(n\,log\,n)$

Combinando, agora, estes dois algoritmos (*FazMonotono* e *TriangulaPoligonoMonotono*) obteremos um algoritmo para a triangulação de polígonos que tem complexidade de tempo $O(n\,log\,n)$ e complexidade de espaço linear. Usaremos, então, como sub-rotina o algoritmo de triangulação para polígonos monótonos (*TriangulaPoligonoMonotono*) para triangular qualquer polígono simples. A ideia é primeiro decompor um

polígono em partes monótonas e depois triangular essas partes. Aparentemente já temos todos os "ingredientes"que precisamos. Há, no entanto, um problema: assumimos anteriormente que a *entrada* era a de um polígono estritamente y-monótono, visto que o algoritmo da subsecção anterior produzia peças monótonas com arestas horizontais. Recordemos que, na subsecção anterior, tratámos de vértices com a mesma ordenada. Este efeito seria o mesmo se fizéssemos uma leve rotação do plano no sentido anti-horário desde que dois vértices não estivessem numa mesma linha horizontal. Acontece que os subpolígonos monótonos produzidos pelo algoritmo (da subsecção anterior), são estritamente monótonos nesta rotação do plano. Assim, o algoritmo de triangulação desta subsecção funciona correctamente se tratarmos os vértices com a mesma ordenada da esquerda para a direita (o que corresponde a trabalhar num plano rodado). Portanto, podemos combinar os dois algoritmos para obter uma triangulação que funciona para qualquer polígono simples.

Este algoritmo de triangulação (combinado) tem tempo $O(n \log n)$ e usa espaço $O(n)$, pois decompor o polígono em partes monótonas leva $O(n \log n)$ (pelo teorema 3.2.1), depois, no segundo passo, triangulámos cada parte monótona com um algoritmo de complexidade linear (feito nesta subsecção). Assim, a soma do número de vértices destas partes monótonas é $O(n)$ e, então, o segundo passo leva $O(n)$ no total. Temos, portanto, o seguinte resultado:

Teorema 3.2.3 *Um polígono simples com n vértices pode ser triangulado em tempo* $O(n \log n)$ *por um algoritmo que usa espaço* $O(n)$.

3.3 Partição de polígonos em polígonos estrelados

Nesta secção consideramos partições em polígonos estrelados e mostramos um algoritmo eficiente proposto por Avis e Toussaint [8]. Uma partição semelhante já havia sido sugerida por Maruyama [72]. A sua solução envolve uma criação de novos pontos de *Steiner*, que produz componentes estreladas sobrepostas e que requer uma complicada e custosa implementação.

A parte principal desta secção, descreve um algoritmo de complexidade $O(n \log n)$ que decompõe um polígono, com n vértices, em polígonos estrelados que não se sobrepõem, ou seja, disjuntos. Esta decomposição não envolve a criação de nenhum novo

vértice e pode produzir sempre uma decomposição com no máximo $\lfloor \frac{n}{3} \rfloor$ polígonos estrelados. No entanto, o número de polígonos estrelados não é o menor na decomposição. Mas por outro lado, este algoritmo, é extremamente flexível e pode facilmente ser modificado para produzir outro tipo de composições completamente diferentes. Este algoritmo segue de perto a prova construtiva de Fisk [35] do teorema de Chvatal [21]: para todo o polígono com n vértices existe uma decomposição disjunta (partição) com no máximo $\lfloor \frac{n}{3} \rfloor$ polígonos estrelados.

3.3.1 Descrição do algoritmo

Antes de descrevermos o algoritmo precisamos de um novo conceito que é o da coloração. Uma **coloração** de uma triangulação, T, é uma atribuição de cores aos vértices de T tal que não existem dois vértices adjacentes com a mesma cor. Claramente, qualquer coloração de T requer pelo menos três cores. De facto, prova-se que três cores são suficientes (indução no número de vértices [78]). Podemos observar, na figura 3.29 a triangulação de um polígono simples com a respectiva coloração representada pela cores $\{1, 2, 3\}$.

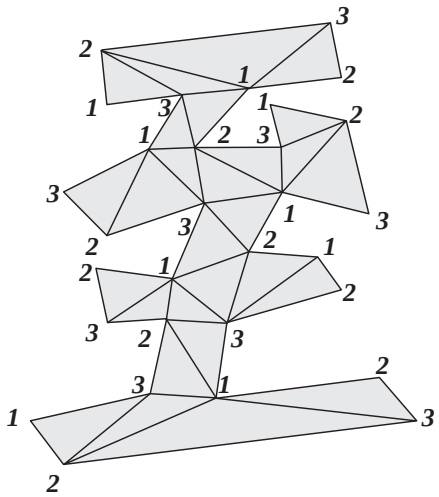


Figura 3.29: Triangulação e coloração de um polígono.

De uma coloração com três cores (ou 3-coloração) de T podemos obter três decomposições diferentes em polígono estrelados de P. Consideremos o conjunto de

vértices que recebem a cor $\{i\}$. Denotemos este conjunto por $S_i = \{s_1, s_2, ..., s_j\}$. P pode ser decomposto em j polígonos estrelados $P_1, P_2, ..., P_j$ onde

$$P_k = \{x \ : \ x \textit{ é um vértice de } P \textit{ e } x \textit{ é adjacente a } s_k \textit{ em } T \ \} \ \cup \ \{s_k\} \ , \ k = 1, 2, ..., j.$$

Além disso, os polígonos estrelados são disjuntos e

$$P = P_1 \cup P_2 \cup ... \cup P_j. \tag{3.1}$$

Consideremos um triângulo $\triangle(x, y, z)$ de T. Pelo menos um destes vértices, digamos, x, recebe a cor $\{i\}$ e está contido em S_i. Suponhamos que $x = s_k$. Então o triângulo $\triangle(x, y, z)$ está contido em P_k. Assim, qualquer triângulo de T está contido nalgum polígono estrelado, verificando-se a equação 3.1.

Claramente que cada uma das três cores usadas para a coloração de T induzem a diferentes decomposições. As três decomposições para a coloração da figura 3.29 estão mostradas nas figuras 3.30 (a), 3.30 (b) e 3.31.

O algoritmo consiste em três partes: a triangulação do polígono, a coloração dos vértices da triangulação com três cores e a construção de polígonos estrelados. O algoritmo está descrito em baixo.

Procedimento *Decompoe*(P)

1. Obtenha uma triangulação T de P.
2. Faça a coloração dos vértices de T com as cores $\{1, 2, 3\}$.
3. Seleccione uma cor arbitrariamente e faça como saída cada vértice com a cor escolhida, conjuntamente com a lista dos vértices adjacentes. (Se pretendermos uma decomposição com no máximo $\lfloor \frac{n}{3} \rfloor$ polígonos estrelados, então deve ser escolhida uma coloração com o mínimo de cores para os vértices.)

O procedimento *Decompoe*(P) é bastante flexível, pois geralmente um polígono tem muitas e diferentes triangulações. É bastante fácil implementar este procedimento num tempo $O(n^2)$, usando variadas estratégias de triangulação. Concluímos esta secção descrevendo uma implementação que requer um tempo $O(n \log n)$.

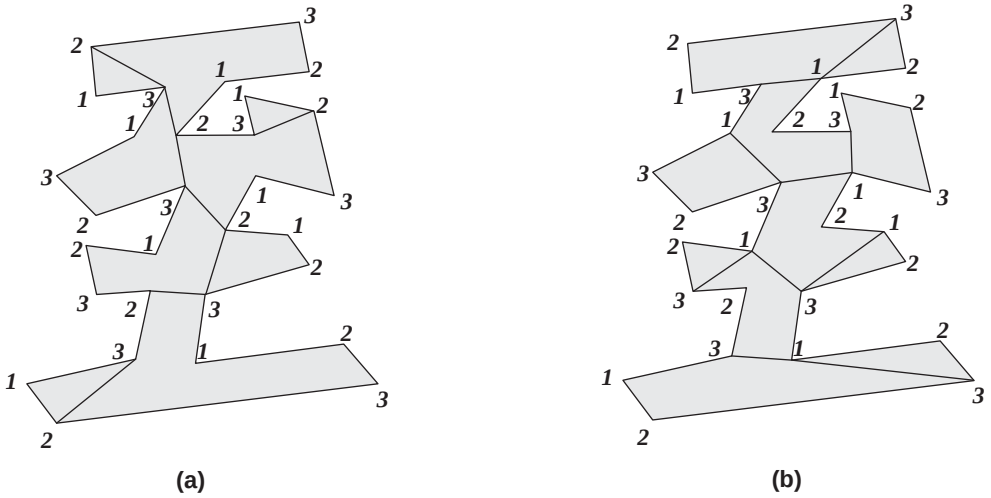


Figura 3.30: (a) Decomposição usando a cor 1.
(b) Decomposição usando a cor 2.

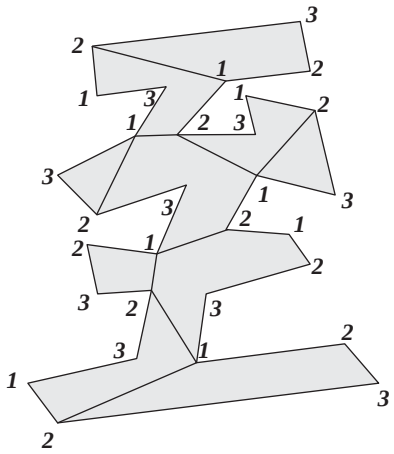


Figura 3.31: Decomposição usando a cor 3.

O 1º passo do procedimento pode ser executado em $O(n \log n)$ se usarmos o algoritmo de Garey et al. [38] descrito na subsecção 3.2.1. O 2º passo também pode ser executado num tempo de complexidade $O(n \log n)$ usando a técnica de"dividir para conquistar"que será descrita mais abaixo. O 3º passo, tal como foi descrito, identifica simplesmente os vértices de cada polígono estrelado, o que requer um tempo $O(n)$. Se

pretendermos, estes vértices podem ser ordenados por uma ordem angular, usando as arestas de T, também num tempo $O(n)$. Se pretendermos uma relação entre os vários polígonos estrelados, podemos inserir uma modificação apropriada no 3º passo. É imediato que cada aresta de T divide P em dois polígonos triangulados mais pequenos. Como pretendemos usar a técnica "dividir para conquistar", procuramos uma aresta que divide T em partes aproximadamente do mesmo tamanho. Usamos depois esta mesma técnica repetidamente nessas partes e, finalmente, unimos as duas partes para darmos uma coloração a P. Este método requer um algoritmo de complexidade $O(n\,log\,n)$ se as duas condições seguintes forem satisfeitas [17]:

(a) No passo de divisão, cada parte deve conter pelo menos uma quantidade fixa, digamos, um quarto dos vértices.

(b) Ambos os passos de divisão e de união requerem tempo $O(n)$.

Façamos um esboço da prova de ambas as condições:

Para a condição (a), consideremos a divisão da fronteira de T em quatro cadeias, A, B, C, D, cada uma com comprimento de pelo menos $\lfloor \frac{n}{4} \rfloor$ (ver figura 3.32). Se houver alguma aresta entre as cadeias A e C a prova está completa, pois esta aresta divide P em duas cadeias com pelo menos $\lfloor \frac{n}{4} \rfloor$ vértices cada uma. Por outro lado, como T é uma triangulação, se não houver nenhuma aresta entre as cadeias A e C, deverá haver arestas entre as cadeias B e D. Qualquer uma das arestas é suficiente para o passo de divisão.
Para a verificação da condição (b), consideremos primeiro o passo de divisão. Este pode ser implementado num tempo $O(n)$, numerando os vértices de P sequencialmente desde 1 até n e nessa altura testa-se cada uma das $O(n)$ arestas para vermos se verificam a condição (a). Isto mostra que o passo de divisão tem complexidade de tempo $O(n)$. Para o passo de união, consideramos as cores que foram designadas para a separação das arestas em cada uma das duas partes. Se as cores forem iguais nas duas partes, o passo de união é trivial. Caso contrário, renomeamos as cores numa das partes de tal modo que as cores coincidam nas arestas de separação. Este processo leva um tempo $O(n)$, portanto a condição (b) é verificada.

Destes exemplos é facilmente observável que as partições podem ser melhoradas por variados processos. Um desses passos poderá ser, remover qualquer aresta que

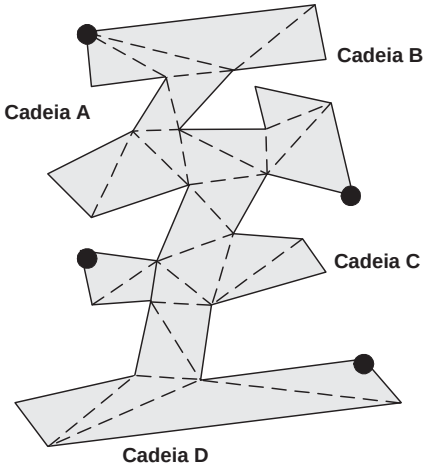


Figura 3.32: Divisão da fronteira de uma triangulação de um polígono simples em quatro cadeias A, B, C e D.

forma um triângulo. Para além disso, diferentes métodos de triangulação podem dar partições melhoradas. Um interessante problema em aberto será encontrar uma decomposição com um número mínimo de polígonos estrelados. Para o caso de decomposições convexas, este problema foi resolvido por Chazelle e Dobkin [17]. Para casos mais especiais, como polígonos ortogonais, a decomposição no número mínimo de rectângulos foi resolvido por Ferrari et al. [34].

3.4 Partição em partes convexas

Para além de algoritmos eficientes para particionar um polígono em triângulos, quadriláteros, partes monótonas, também é de interesse o desenvolvimento de algoritmos que particionem um polígono em polígonos convexos. Uma possível motivação para particionarmos um polígono em partes (polígonos) convexas, é o reconhecimento de caracteres: um caracter pode ser representado como um polígono particionado em partes convexas.

A partição de polígonos em triângulos pode ser vista como um caso particular do problema de particionarmos um polígono em partes convexas. Ao particionarmos um polígono em partes convexas estaremos interessados em (1) minimizar o número

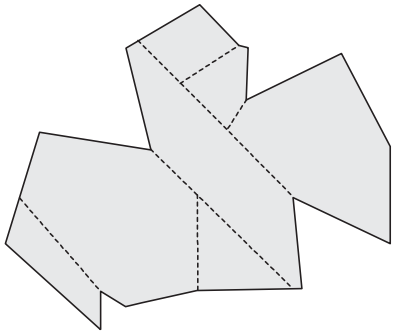


Figura 3.33: Polígono particionado em partes convexas.

de partes e (2) construir esta partição o mais rápido possível. Estes dois objectivos obviamente entram em "conflito".

3.4.1 Partição óptima

Para analisarmos a qualidade (em termos de número de partes) de uma partição é útil conhecermos limites inferiores e superiores para o menor número possível de partes em que o polígono pode ser particionado.

Teorema 3.4.1 (Chazelle) *Seja Φ o menor número de partes convexas que um polígono com r vértices reflexos pode ser particionado. Então, temos que: $\lceil \frac{r}{2} \rceil + 1 \leq \Phi \leq r + 1$.*

Prova: A prova de que $\Phi \leq r + 1$ é algorítmica.

Algoritmo Particiona (P).

Entrada: um polígono $P = (v_0, ..., v_{n-1})$.

Saída: uma partição de P por segmentos em subpolígonos convexos.

1. Se P não tem vértices reflexos, então pare. [Neste caso, P já é convexo].
2. Seja v um vértice reflexo de P.
3. Trace um segmento $s \subset P$ tendo v como uma das extremidades e a outra extremidade em ∂P que elimine este vértice reflexo.

4. Aplique o algoritmo recursivamente aos dois polígonos formados.

O número de partes convexas produzidas pelo algoritmo acima é no máximo $r+1$ (ver figura 3.34). Este algoritmo desenha um segmento que bissecta cada ângulo reflexo, removendo, desta forma, todos os ângulos desta amplitude. Fica, assim, criada uma partição convexa.

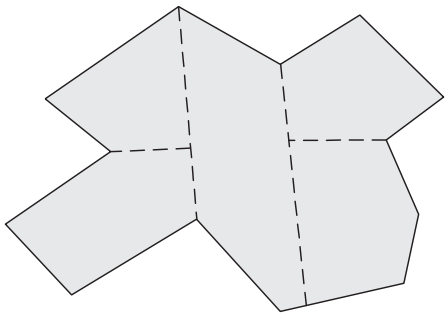


Figura 3.34: O algoritmo criou $r+1$ partes convexas: $r = 4$; 5 peças.

Todos os ângulos reflexos precisam de ser destruídos para produzirmos partes convexas. No máximo dois desses ângulos podem ser eliminados pela inclusão de um único segmento. Logo $\Phi \geq \lceil \frac{r}{2} \rceil + 1$ (ver figura 3.35). ■

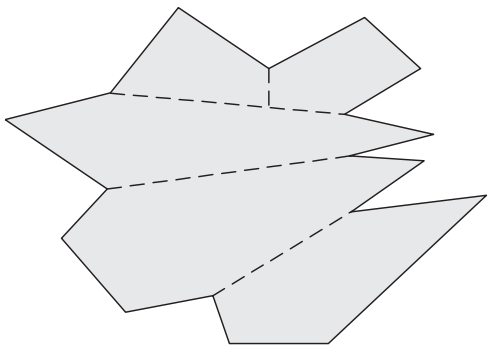


Figura 3.35: O algoritmo criou $\lceil \frac{r}{2} \rceil + 1$ partes convexas: $r = 7$; 5 peças.

3.4.2 O algoritmo de Hertel e Mehlhorn

Hertel e Mehlhorn [45] desenvolveram um algoritmo de aproximação de complexidade de tempo linear para particionar um polígono em partes convexas por diagonais. Numa partição convexa por diagonais de um polígono, diremos que uma **diagonal**, d, é **essencial** para um vértice v, se a remoção de d cria uma parte que é não-convexa em v (v é um vértice reflexo e d é incidente a v).

O algoritmo de Hertel e Mehlhorn consiste em:

- construir uma triangulação de P;
- remover arbitrariamente diagonais não-essenciais até que todas as diagonais restantes sejam essenciais.

Este algoritmo particiona um polígono em partes convexas num tempo linear (se usarmos o algoritmo de Chazelle [16] para a triangulação de polígonos).

Lema 3.4.1 *Numa partição convexa, por diagonais, existem no máximo duas diagonais essenciais para cada vértice reflexo.*

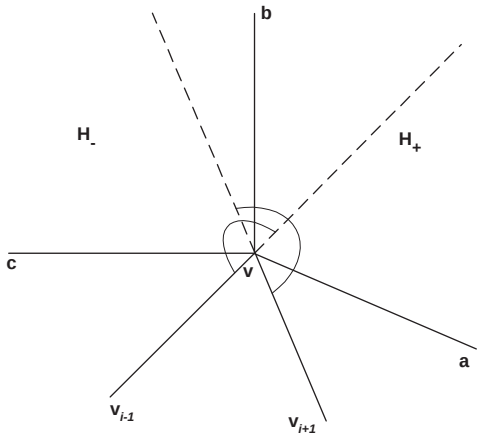


Figura 3.36: A diagonal a é não essencial, pois b também está em H_+. Analogamente, c é não essencial.

Prova: Seja v um vértice reflexo e v_{i-1} e v_{i+1} os seus vértices adjacentes. Existe no máximo uma diagonal essencial no semiplano H_+ à esquerda $\overline{vv_{i+1}}$; se existirem duas, a que está mais perto de $\overline{vv_{i+1}}$ pode ser removida sem que criemos um vértice não-convexo em v (ver figura 3.36). Analogamente, existe no máximo uma diagonal essencial ao semiplano H_- à esquerda de $\overline{v_{i-1}v}$. Juntos, esses semiplanos cobrem o ângulo interior em v. Logo existem no máximo duas diagonais essenciais para v. ■

Teorema 3.4.2 *O algoritmo de Hertel-Mehlhorn dá-nos uma partição que nunca tem mais que quatro vezes o número de partes convexas de uma partição óptima por diagonais (isto é, uma partição por diagonais com o número mínimo de partes).*

Prova: Quando o algoritmo pára, toda a diagonal é essencial para algum vértice (reflexo). Pelo lema 3.4.1, cada vértice reflexo pode ser responsável por no máximo duas diagonais essenciais. Logo, o número de diagonais essenciais não pode ser maior que $2r$, onde r é o número de vértices reflexos (pode ser menor se alguma diagonal for essencial para ambos os seus vértices extremos). Assim, o número M de partes convexas produzidas pelo algoritmo satisfaz $2r + 1 \geq M$. Como $\Phi \geq \lceil \frac{r}{2} \rceil + 1$ pelo lema 3.4.1, então, $4\Phi \geq 2r + 4 > 2r + 1 \geq M$. ■

Partição convexa óptima: o algoritmo para encontrar uma partição óptima de um polígono em partes convexas é devido a Greene [42], 1983. A sua complexidade é $O(r^2\, n^2) = O(n^4)$. Este algoritmo foi melhorado, em 1985, por Keil [50] que desenvolveu um algoritmo de complexidade $O(r^2\, n\, log\, n) = O(n^3\, log\, n)$ (ambos utilizaram técnicas de programação dinâmica).

Se a partição puder ser formada por segmentos arbitrários o problema é mais difícil ainda (a partição pode usar segmentos cujos extremos não intersectam a fronteira do polígono - os pontos de Steiner (ver figura 3.37)). Apesar dessa dificuldade acrescida, em 1980, na sua tese de doutoramento, Chazelle [15] resolveu este problema propondo um algoritmo de complexidade $O(n + r^3) = O(n^3)$.

3.5 Quadrangulação

Nos últimos tempos, tem-se vindo a chegar à conclusão que as quadrangulações têm muitas vantagens relativamente às triangulações. Por exemplo, para o problema da

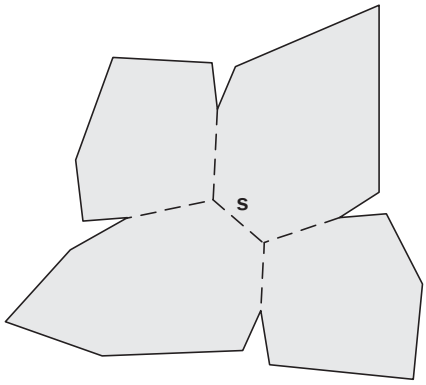


Figura 3.37: Uma partição convexa óptima. O segmento s não toca ∂P.

interpolação de dados dispersos [58] e nos melhoramentos na análise da elasticidade, que pode ser obtida por métodos de elementos finitos, é preferível o uso de quadrangulações às triangulações [4]. Infelizmente não se sabe muito sobre teoria de quadrangulações de um conjunto de pontos e boas redes quadrangulares são mais difíceis de gerar que boas redes triangulares [47]. De facto, se somente for permitido inserir diagonais entre um dado conjunto de pontos, isto é, se não forem permitidos pontos de Steiner, então nem todos os conjuntos de pontos admitem uma quadrangulação. A caracterização de quadrangulações de conjunto de pontos e a construção de algoritmos para uma eficiente computação usando um número mínimo de pontos de Steiner é um assunto bastante recente [12]. Em [12] é mostrado que um conjunto de pontos admite uma quadrangulação não usando pontos de Steiner se e só se o número de pontos do invólucro convexo for ímpar. A partir do momento que se começou a dar mais atenção à computação de quadrangulações, visto que as triangulações têm sido estudadas há várias décadas [10], começou-se a investigar o problema de converter triangulações em quadrangulações [44, 48, 98]. No entanto, estes métodos são heurísticos, conceptualmente incómodos e recorrem a demasiados pontos de Steiner. Por exemplo, Johnston et al. [48] integrou várias heurísticas num sistema que automaticamente converte uma rede triangular numa quandrangular, com complexidade $O(n^2)$ e pode inserir mais do que n pontos de Steiner no processo, onde n representa o número de redes triangulares. Nenhuma tentativa foi feita tanto para optimizar o número de pontos de Steiner ou a complexidade dos algoritmos. Notemos que a quadrangulação de polígonos (sem pontos no seu interior) tem sido investigada na literatura de geometria computacional em vários contextos.

Atentemos que, tal como um conjunto de pontos do plano, a quadrangulação de polígonos simples nem sempre é possível. No entanto, não é difícil construir polígonos que requerem $\Omega(n)$ pontos de Steiner por forma a completar a quadrangulação. Por outro lado, os polígonos ortogonais, admitem sempre uma quadrangulação sem recorrer a pontos de Steiner. De facto, tais polígonos, admitem sempre quadrangulações em que cada quadrilátero é convexo. Por esta razão, quadrangulações não convexas de polígonos ortogonais não têm interesse não tendo sido, por isso, objecto de estudo. Uma primeira prova existencial de que os polígonos ortogonais admitem sempre quadrangulações convexas, foi dada por Klawe e Kleitman [49]. Uma prova construtiva com um algoritmo de complexidade $O(n)$ foi primeiramente obtido por Sack e Toussaint [92] para polígonos estrelados subsequentemente generalizado para correr num tempo $O(n \log n)$ para um polígono ortogonal arbitrário por Sack [91]. Edelsbrunner, O´Rourke e Weltz [27], Lubiw [70], Sack e Toussaint [93], entre outros, mais tarde, obtiveram outras provas construtivas com complexidades semelhantes.

3.5.1 Quadrangulação de polígonos triangulados

Como já foi referido, nem todos os polígonos admitem uma quadrangulação. Nestes casos, é necessário adicionar pontos de Steiner, para o quadrangularmos. Nesta subsecção, veremos como obter uma quadrangulação após o polígono estar triangulado. Isto implica que poderemos apagar diagonais existentes, mas não poderemos inserir novas diagonais entre pares de vértices. Também não poderemos eliminar vértices de polígono original.

Provavelmente, o método mais simples de quadrangular um polígono após este estar triangulado, é inserir primeiro um ponto de Steiner no interior de cada aresta e diagonal do polígono triangulado. Então, para cada triângulo inserimos um ponto de Steiner extra em qualquer sítio do interior do triângulo (de tal modo que não fiquem três pontos de Steiner colineares com qualquer outro par de pontos de Steiner nesse triângulo) e ligamo-lo aos três outros pontos de Steiner desse triângulo. Uma quadrangulação desse tipo está ilustrada na figura 3.38.

Este método tem várias vantagens, uma das quais, se o ponto de Steiner for bem colocado no interior do triângulo definido pelos outros três pontos de Steiner, pode-se obter uma quadrangulação convexa [31]. Este algoritmo é fácil de implementar

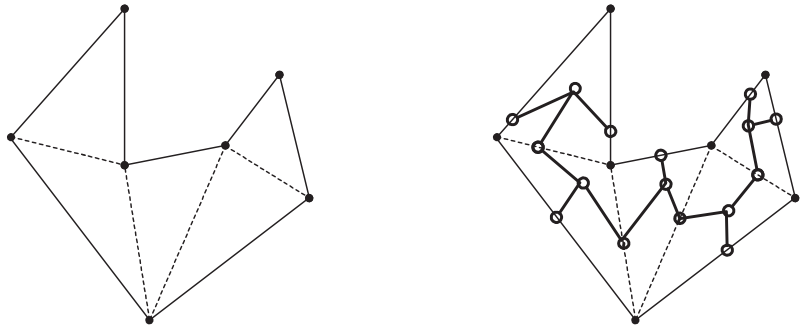


Figura 3.38: Exemplo de uma construção de uma quadrangulação a partir de uma triangulação.

e tem complexidade linear. O problema com este método é que, apesar de levar a quadrangulações rigorosas, usa muitos pontos de Steiner, quando é desejável que este número de pontos seja o mais reduzido possível. De facto, este método usará sempre $3n - 5$ pontos de Steiner num polígono simples triangulado com n vértices.

Um outro método que usa aproximadamente um terço dos pontos de Steiner é o algoritmo de triangulação Hamiltoniano de Arkin et al. [5]. Com um objectivo diferente em mente, nomeadamente, uma prestação mais eficaz na área da computação gráfica, Arkin et al. propuseram um método elegante de obter ao que eles chamaram uma triangulação do ciclo Hamiltoniano. Bose e Toussaint [12], propuseram um método para obter uma quandragulação de um conjunto de pontos via, ao que eles denominaram, uma triangulação serpentina. Uma triangulação é serpentina se o seu grafo dual admite um caminho Hamiltoniano. Combinando as ideias em [12] e [5] podemos obter um algoritmo para quadrangular um polígono simples triangulado como se segue (ver figura 3.39).

Primeiro a triangulação do ciclo Hamiltoniano é obtida com o algoritmo de Arkin et al. [5]. Consideremos uma triangulação de um polígono simples como na figura 3.39 (a). Primeiro, insere-se uma árvore dual planar no polígono triangulado. Esta inserção pode ser sempre feita numa triangulação ou numa quadrangulação convexa e foi provada primeiro por Bern e Gilbert [11]. Depois, em cada triângulo, o nó da árvore dual correspondente ao triângulo é ligado com arestas aos seus três vértices. Finalmente, as diagonais originais do polígono triangulado são removidas para permitir a triangulação de Hamilton, mostrada na figura 3.39 (c). O ciclo de Hamilton contido no

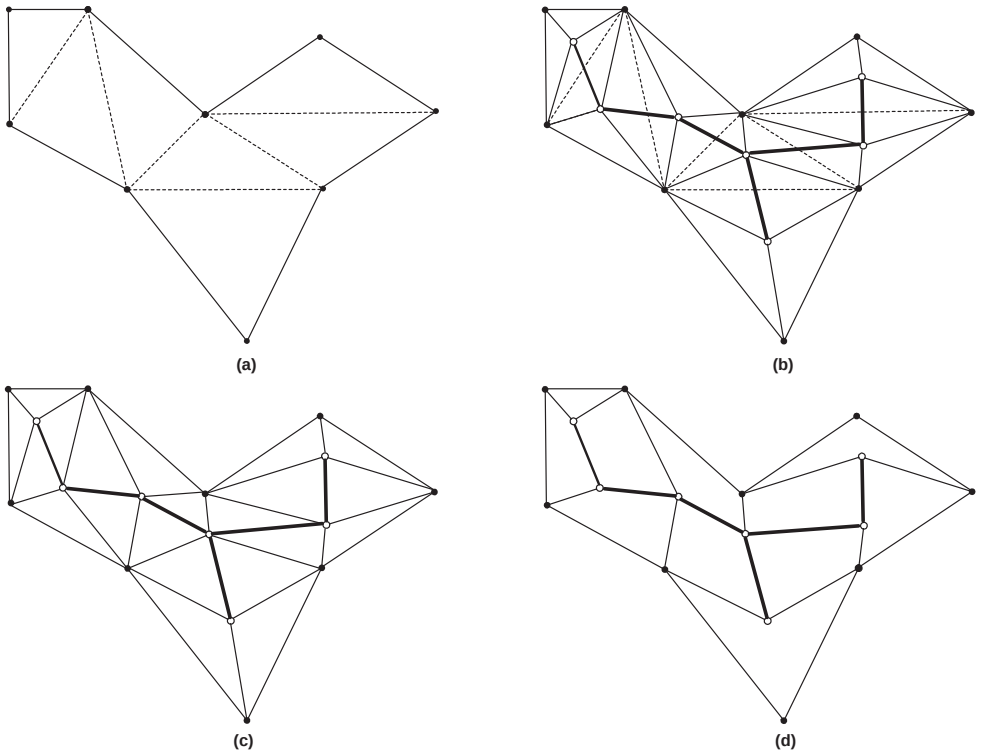


Figura 3.39: Quadrangulação via uma triangulação de Hamilton. (a) Polígono original triangulado. (b) Árvore dual geométrica com cada nó da árvore ligada aos três vértices dos seus triângulos correspondentes. (c) Eliminadas as diagonais originais. (d) Uma quadrangulação resultante com um triângulo que sobra, onde é inserido um ponto exterior de Steiner.

dual da triangulação pode ser encontrado executando uma árvore transversal da árvore dual geométrica; isto permite-nos visitar cada triângulo na ordem Hamiltoniana. Para obter uma quadrangulação é suficiente seguir a ordem Hamiltoniana (começando num triângulo qualquer) e eliminando cada uma das outras diagonais. Uma quadrangulação obtida por esta forma, está ilustrada na figura 3.39 (d). Notemos que o último elemento pode ser um triângulo e, nesse caso, podemos juntar um ponto de Steiner adicional (fora do polígono) para convertermos este triângulo num quadrilátero. No entanto este algoritmo é ligeiramente mais complicado que o anterior, continuando no entanto, a ter complexidade $O(n)$. Além disso, é necessário no máximo um ponto de Steiner (fora do

polígono) e o número de pontos de Steiner do interior do polígono é sempre $n-2$, isto é, no máximo $n-1$ pontos de Steiner no total. Notemos que este método não viola as condições de conversão de uma triangulação para a quadrangulação porque mesmo que descarte todas as diagonais, o processo não insere novas diagonais entre pares de vértices. Embora a aproximação de Hamilton melhore o número de pontos de Steiner usados, mostra-se que, usando argumentos de coloração para triangulação de polígonos [21, 35], podemos reduzir ainda mais o número de pontos de Steiner para cerca de um terço e esse número é optimal.

Antes de continuarmos, definamos pontos de Steiner com maior precisão. Como já foi referido, nenhum ponto de Steiner poderá estar sobre a fronteira do polígono ou numa diagonal. Portanto, consideramos apenas dois pontos de Steiner: interiores e exteriores. Pontos de Steiner interiores estão estritamente no interior do polígono, mas não em diagonais e pontos de Steiner exteriores, estão no exterior do polígono. Além disso, para o caso em que somente são permitidos pontos de Steiner exteriores, a fronteira do polígono original poderá ser modificada da seguinte maneira: cada ponto de Steiner p está associado com uma aresta e da diagonal do polígono, a aresta e é eliminada e duas novas arestas são criadas ligando p aos novos pontos extremos de e.

O teorema seguinte dá-nos limites para o número de pontos de Steiner que são necessários para a quadrangulação de um polígono triangulado segundo certas condições:

Teorema 3.5.1 $\lfloor \frac{n}{3} \rfloor$ *pontos de Steiner são sempre suficientes, e algumas vezes necessários, para quadrangular um polígono simples triangulado com n vértices. Além disso, estes pontos de Steiner podem ser localizados em tempo* $O(n)$.

Prova: Fisk [35] observou que desde que os vértices da triangulação do polígono possam ser coloridos com três cores, então toda a triangulação de um polígono com n vértices pode ser particionado em $\leq \lfloor \frac{n}{3} \rfloor$ *leques* (um *leque* é uma triangulação que tem um vértice, chamado *centro do leque*, que é partilhado por todos os triângulos). Observemos que há sempre uma partição, tal que esses *leques* começam e acabam numa aresta do polígono (isto advém do argumento da coloração de três cores, usado por Fisk para particionar um polígono triangulado em *leques*). Chamamos a tais arestas de P *braços do leque* (ver figura 3.40). Cada *braço do leque* aparece somente num *leque*.

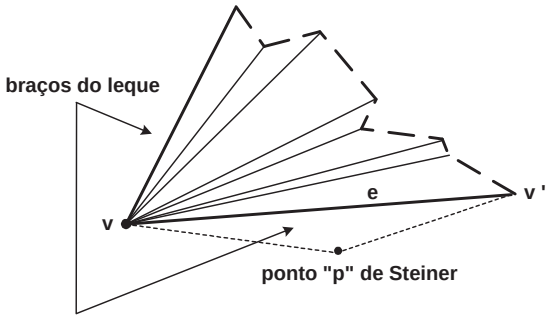


Figura 3.40: Um *leque* na partição, começa e acaba sempre numa aresta de um polígono.

Consideremos agora um vértice, v, de P que é um *centro do leque*. O vértice v define uma sequência de triângulos na triangulação. Estes triângulos podem ser emparelhados para formarem quadriláteros. Se o número desses triângulos for ímpar, sobrará um triângulo e uma dessas arestas é um *braço do leque*, e. Um dos pontos extremos de e é v; seja v' o outro ponto extremo. Podemos converter esta situação para um quadrilátero adicionando um ponto de Steiner p numa posição conveniente, não pertencente a e, eliminando a aresta e e ligando p ao dois vértices, v e v'.

Assim, precisamos de adicionar no máximo um ponto de Steiner por *leque*. Consequentemente, P pode ser particionado em $\leq \lfloor \frac{n}{3} \rfloor$ *leques*, e teremos, então, que $\lfloor \frac{n}{3} \rfloor$ pontos exteriores de Steiner serão sempre suficientes para quadrangular um polígono simples triangulado.

Para mostrarmos que $\lfloor \frac{n}{3} \rfloor$ pontos exteriores de Steiner são algumas vezes necessários para quadrangular um polígono triangulado, consideremos o polígono triangulado da figura 3.41 (este exemplo é semelhante a um exemplo de um polígono simples que necessite de $\lfloor \frac{n}{3} \rfloor$ guardas). Há apenas três maneiras para escolher os *leques*:

- se v_1 é escolhido como *centro do leque*, então os outros *centros dos leques* deverão ser os vértices $v_4, v_7, v_{10}, ..., v_{n-2}$. Estes *centros* formam triângulos e, por esta razão, cada um precisará de um ponto exterior de Steiner para a quadrangulação.

- Se $v_3, v_6, v_9, ..., v_{n-3}, v_n$ forem escolhidos como *centros dos leques*, cada um dos *centros* tem um número ímpar de triângulos, e por esta razão, cada um deles precisará de um ponto exterior de Steiner para a quadrangulação.

- Se $v_2, v_5, v_8, ..., v_{n-1}$ forem escolhidos como os *centros dos leques*, teremos um caso similar ao ponto anterior.

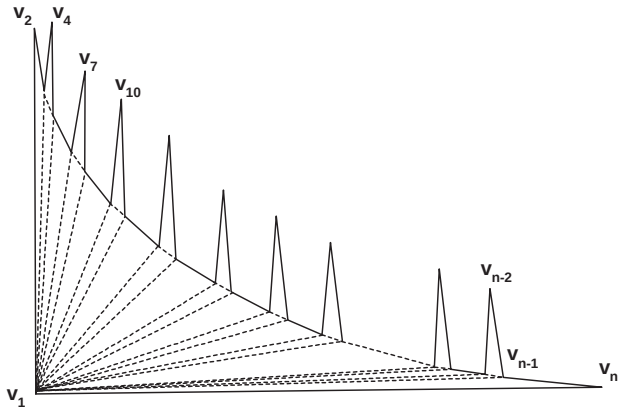


Figura 3.41: Uma quadrangulação deste polígono requer $\lfloor \frac{n}{3} \rfloor$ pontos exteriores de Steiner. Este polígono admite apenas uma triangulação (como é mostrada).

Vemos que em cada um dos casos descritos acima, são necessários $\lfloor \frac{n}{3} \rfloor$ pontos exteriores de Steiner para obter uma quadrangulação a partir de um polígono triangulado.
Para mostrarmos que estes pontos de Steiner podem ser localizados num tempo $O(n)$, consideremos o seguinte: o polígono triangulado pode ser colorido com três cores num tempo linear (Kooshesh e Moret [60]). A aresta onde um guarda é situado dá-nos o *braço do leque*, e, onde iremos colocar o ponto exterior de Steiner. Para encontrar o lugar adequado para o ponto, podemos triangular o polígono (ou polígonos) que estão fora de P e dentro do invólucro convexo de P, num tempo $O(n)$ usando o algoritmo de Chazelle [16]. O ponto de Steiner para e pode ser colocado num sítio qualquer dentro do triângulo incidente em e (e no exterior de P). Se e for uma aresta do invólucro convexo, então o ponto de Steiner pode ser localizado no interior da região determinada pela intersecção de três planos: um determinado pela aresta e em questão e que não contém P, e os outros dois determinados pelas arestas do invólucro convexo adjacente a e e

que contém P. Então todos os pontos de Steiner podem ser colocados num tempo $O(n)$. ■

O teorema 3.5.1 implica um resultado mais importante no que concerne à quadrangulação de polígonos simples em geral, isto é, não estando o polígono triangulado. Primeiro, dado um polígono simples, este pode ser sempre triangulado num tempo $O(n)$ [16] antes da aplicação do algoritmo de conversão. Segundo, o polígono da figura 3.41 admite somente uma possível triangulação (como é mostrado) e, se não forem permitidos pontos interiores de Steiner, somente estas diagonais podem ser usadas na quadrangulação do polígono. Temos, então, o seguinte resultado:

Corolário 3.5.1 *$\lfloor \frac{n}{3} \rfloor$ pontos de Steiner são sempre suficientes, e alguma vezes necessários, para quadrangular qualquer polígono simples com n vértices. Além disso, estes pontos de Steiner podem ser localizados em tempo $O(n)$.*

3.5.2 Pontos interiores de Steiner e quadrangulações de polígonos triangulados.

Nesta subsecção introduz-se um algoritmo, que chamamos filtração-Q, que converte um polígono triangulado para uma quadrangulado enquanto adiciona pontos de Steiner no interior do polígono, com no máximo um ponto exterior de Steiner. Notemos que nem sempre podemos evitar adicionar pontos exteriores de Steiner, isto é, há polígonos que não podem ser quadrangulados somente com pontos interiores de Steiner. Assim, um n-ágono tem exactamente $n+2s-2$ triângulos numa qualquer triangulação com s pontos interiores de Steiner, donde temos imediatamente que, pontos interiores de Steiner sozinhos não serão suficientes quando n é ímpar (este facto é também usado em [12]). Pontos interiores de Steiner são uma importante consideração quando o objectivo é quadrangular um polígono simples sem modificar a fronteira do polígono. Definamos, agora, com maior precisão pontos interiores de Steiner. Tal como os pontos exteriores de Steiner, é permitido a eliminação de diagonais do triângulo original e não é permitido adicionar novas diagonais entre vértices do polígono. Só é permitido a adição de diagonais entre pontos interiores de Steiner e vértices do polígono.

Primeiro consideremos uma versão mais simples do algoritmo, que nos dá um limite superior de $\lfloor \frac{n}{2} \rfloor$ pontos interiores de Steiner (e no máximo um ponto exterior

de Steiner) para quadrangular um polígono simples triangulado. Seja T a árvore dual do polígono simples triangulado, que assumimos ter raiz num nó de grau 1 e seja h o número de níveis de T (consideramos a raiz o nível 1). O algoritmo de filtração-Q começa no nível mais baixo de T e vai eliminando a árvore à medida que a percorre até ao topo. Seja V_h o conjunto de nós do nível h de T. Seja $v \in V_h$ e seja $par(v)$ o parente de v. Temos, então, os seguintes casos:

Caso 0: Se $par(v)$ for um nó de grau 1, então v e $par(v)$ (isto é, o triângulo correspondentes a estes nós) formam um quadrilátero. Remova estes dois nós de T. Se $par(v)$ for NIL, então teremos simplesmente um triângulo que pode ser quadrangulado com um ponto exterior de Steiner. Notemos que este é o único ponto exterior de Steiner adicionado neste método. Remova v de T.

Caso 1: Se $par(v)$ é um nó de grau 2, então v e $par(v)$ formam um quadrilátero. Remova estes dois nós de T.

Caso 2: Se $par(v)$ (chamemos-lhe u) for um nó de grau 3, então seja w um irmão[10] de v. Então, como está ilustrado na figura 3.42, podemos adicionar um ponto, p, de Steiner no triângulo $\triangle_u$ correspondente ao nó u. Liguemos p aos três vértices de $\triangle_u$, dividindo-o, assim, em três triângulos mais pequenos, $\triangle_{u1}$, $\triangle_{u2}$ e $\triangle_{u3}$ tal que $\triangle_{u2}$ é adjacente ao triângulo $\triangle_v$ e $\triangle_{u3}$ adjacente ao triângulo $\triangle_w$. Assim, os triângulos $\triangle_v$ e $\triangle_{u2}$ podem ser emparelhados por forma a formarem um quadrilátero, assim como os triângulos $\triangle_w$ e $\triangle_{u3}$. Agora na árvore T, elimine-se os nós v e w. O nó u corresponde agora ao triângulo $\triangle_{u1}$.

Após o passo acima ter sido executado para todos os nós em V_h, continuamos com o conjunto de nós no nível mais baixo "da nova versão" da árvore T. Este passo é repetido sucessivamente nas "novas" árvores até termos uma árvore vazia. O conjunto de nós em cada um dos níveis de T pode ser mantido num conjunto de listas ligadas. Observemos que todos os pontos de Steiner (excepto possivelmente um) são adicionados no interior do polígono. Além disso, o número de pontos exteriores de Steiner adicionado é igual ao número de triângulos da triangulação que correspondem a nós de grau três na árvore dual T. Assim, dois nós são eliminados cada vez que um ponto de Steiner é adicionado, pelo que no pior caso este algoritmo adiciona no máximo $\lfloor \frac{n}{2} \rfloor$ pontos interiores de Steiner e no máximo um ponto exterior de Steiner.

[10] Considera-se irmão quando tem os mesmos pais.

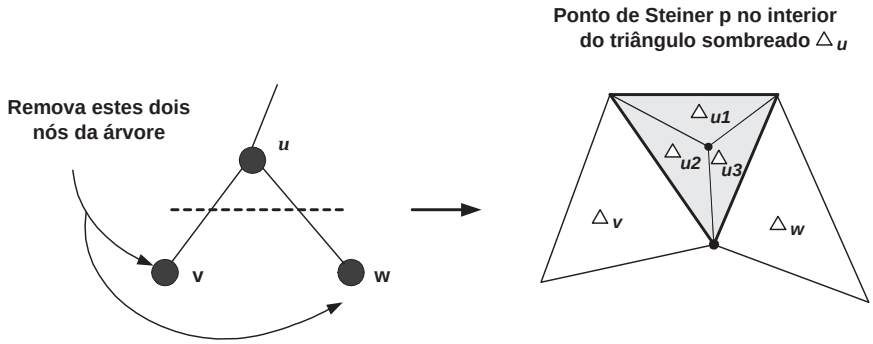


Figura 3.42: Um ponto de Steiner p pode ser adicionado dentro do triângulo $\triangle_u$ correspondente a um nó de grau 3 na árvore dual, como está mostrado à direita.

Este método adiciona pontos de Steiner de uma forma conservadora, ou seja, podemos ir apertando o limite superior explorando a estrutura da árvore T. Mostraremos agora que é possível eliminar pelo menos quatro nós de T cada vez que um ponto interior de Steiner for adicionado. Para provarmos o limite superior, usamos a propriedade que os pentágonos são estrelados desde algum ponto do seu interior.

Teorema 3.5.2 *O seguinte algoritmo de filtração-Q executa uma quadrangulação de um n-ágono triangulado com no máximo* $\lfloor \frac{n}{4} \rfloor$ *pontos interiores de Steiner e no máximo um ponto exterior de Steiner num tempo* $O(n)$.

Prova: Analisaremos os seguintes casos, onde V_h significa o conjunto de nós do nível h de T.

Passo 1: Faça o seguinte para cada nó $v \in V_h$: se v é tal que $par(v)$ é NIL então teremos simplesmente um triângulo que pode ser quadrangulado com um ponto exterior de Steiner. Apagamos v de T. Se $par(v)$ for um nó de grau 1, então estes dois nós correspondem a um quadrilátero e apagamos v e $par(v)$ de T. Para os restantes nós $v \in V_h$ tal que $par(v)$ é um nó de grau 2, apague-se v e $par(v)$ de T (v e $par(v)$ formam um quadrilátero) e actualize o grau do parente de $par(v)$.

Passo 2: Se V_h não estiver vazio, faça o seguinte para cada $v \in V_h$: (note que todos os vértices v em V_h são tais que $par(v)$ é um nó de grau 3. Seja w um irmão de v. Referente à figura 3.43, a linha mais fina a tracejado, indica a parte de T que foi apagada neste passo e o triângulo sombreado refere-se à região onde o polígono, possivelmente, continua. Assumimos que indenpendentemente dos nós serem apagados de T, o grau de um nó afectado é actualizado apropriadamente. Aplica-se um dos seguintes casos:

Caso1: $par(par(v))$ é um nó de grau 1 ou 2 (ver figura 3.43 (a)). Seja o triângulo $\triangle_{abc}$ o correspondente a $par(v)$. Neste caso, adicionamos um ponto p de Steiner no interior do triângulo $\triangle_{abc}$ de tal maneira que não estejam três pontos colineares com qualquer dos vértices dos quatro triângulos em questão. Insira as diagonais $[pa]$, $[pb]$ e $[pc]$, formando três quadriláteros: a união dos triângulos $\triangle_{pab}$ e $\triangle_v$, a união dos triângulos $\triangle_{pbc}$ e $\triangle_w$ e a união dos triângulos $\triangle_{pac}$ e o triângulo correspondente a $par(par(v))$. Elimine v, w, $par(v)$ e $par(par(v))$ de T.

Caso2: $par(par(v))$ é um nó de grau 3. Observe que devido ao passo 1, o irmão de $par(v)$ tem que ser um nó de grau 1 ou de grau 3. Assim teremos os seguintes dois sub-casos:

Caso 2.1 O irmão de $par(v)$ é um nó de grau 1 (ver figura 3.43 (b)). Os cinco triângulos correspondentes aos cinco nós em questão são convertidos em três quadriláteros e um triângulo, da seguinte maneira: seja $[abcd]$ o quadrilátero formado pela união dos triângulos $\triangle_{abc}$ e $\triangle_{acd}$, correspondendo, respectivamente, a $par(v)$ e $par(par(v))$. Elimine a diagonal $[ac]$. O quadrilátero $[abcd]$ é estrelado (pelo menos desde qualquer ponto no interior do segmento $[ac]$). Consideremos um ponto, p, de Steiner no interior do núcleo de $[abcd]$ de tal modo que não sejam criados três pontos colineares com os vértices dos triângulos em questão, incluindo o parente de $par(par(v))$. Insira diagonais desde p até a, b, c e d criando quatro novos triângulos sendo p o vértice do topo e os lados de $[abcd]$ como bases. Elimine-se, agora, as diagonais $[ab]$, $[bc]$ e $[cd]$ para formar três novos triângulos. O triângulo $\triangle_{pad}$ é agora o novo triângulo correspondente ao $par(par(v))$. Elimine v, w, $par(v)$ e o irmão de $par(v)$ de T. O nó $par(par(v))$ representa agora o triângulo mais pequeno obtido pela adição de quatro diagonais.

Caso 2.2 O irmão de $par(v)$ é um nó de grau 3 (ver figura 3.43 (c)). Os sete triângulos correspondentes aos sete nós em questão são convertidos em quatro quadriláteros e um triângulo da seguinte maneira: seja $[abcde]$ o pentágono formado pela união dos três triângulos $\triangle_{abc}$, $\triangle_{cde}$ e $\triangle_{ace}$ correspondentes, respectivamente, ao $par(v)$, ao irmão de $par(v)$, e $par(par(v))$. Elimine as diagonais $[ac]$ e $[ce]$. O pentágono $[abcde]$ é estrelado desde uma dada região do seu interior. Considere-se um ponto p de Steiner no interior do núcleo do pentágono $[abcde]$ tal que não se formem três pontos colineares com qualquer dos vértices dos triângulos em questão, incluindo o parente de $par(par(v))$. Insira diagonais desde p até a, b, c, d e e, criando cinco novos triângulos, com p como vértice do topo e os lados do pentágono $[abcde]$ as suas bases. Elimine agora as diagonais $[ab]$, $[bc]$, $[cd]$ e $[de]$ para formarem quatro novos quadriláteros. O triângulo $\triangle_{pae}$ é o novo triângulo correspondente ao $par(par(v))$ em T. Elimine agora os seguintes nós de T: v, w, $par(v)$, o irmão de $par(v)$ e dois filhos (folhas) deste nó.

Repita os passos 1 e 2 na versão "podada"de T e continue até a restante árvore ficar vazia. Observe que cada vez que o algoritmo de filtração-Q adiciona um ponto interior de Steiner, pelo menos quatro nós são removidos de T. No último passo antes da árvore ficar vazia, um ponto exterior de Steiner será adicionado. ■

Foram apresentados algoritmos eficientes para converter domínios triangulados para quadrangulações e foram dados limites de pontos de Steiner que podem ser necessários para obter essas quadrangulações. Mostrou-se que, em tempo linear, um n-ágono simples triangulado pode ser quadrangulado com um número mínimo possível de pontos exteriores de Steiner necessários para essa triangulação. Mostrou-se que são suficientes $\lfloor \frac{n}{3} \rfloor$ pontos de Steiner e, alguma vezes necessários, para quadrangular n-ágonos simples. Também se mostrou que, são suficientes $\lfloor \frac{n}{4} \rfloor$ pontos interiores de Steiner (e no máximo um ponto exterior de Steiner) para quadrangular um n-ágono simples triangulado e este processo pode ser executado em tempo linear.

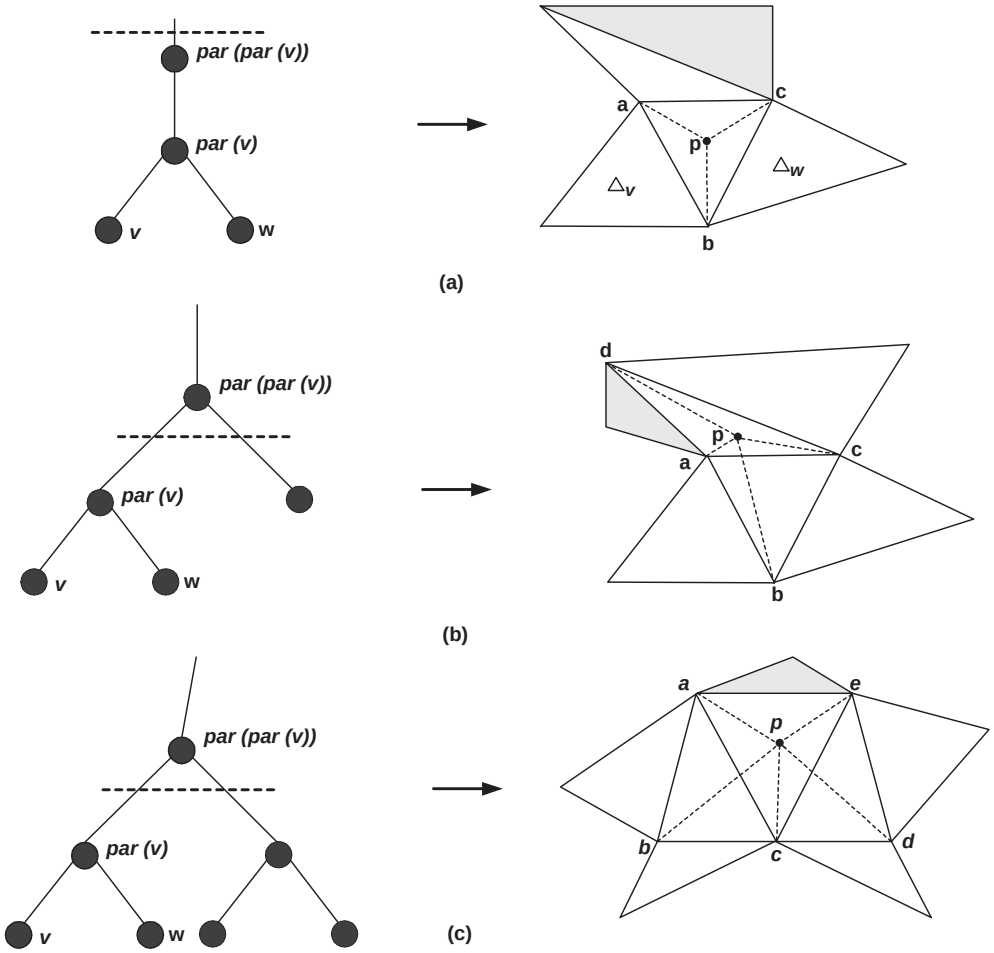


Figura 3.43: Os três casos que podem surgir no algoritmo de filtração-Q.

Capítulo 4

Partição não clássica de polígonos

Pseudo-triângulos e pseudo-triangulações têm recebido nos últimos anos uma particular atenção devido às suas aplicações. Originalmente em contextos como a visibilidade [83, 84, 87, 89] e *ray shooting*[1] [2, 83, 56]; somente em aplicações recentes como a detecção cinética de colisões [55, 1, 84], planeamento de movimentos de *robots* [99] e movimento rígido [23, 99] fizeram com que crescesse o interesse pelo estudo das suas propriedades combinatórias e geométricas [99]. Há, no entanto, ainda várias questões em aberto relacionadas com a pseudo-triangulação [100].

Um **pseudo-triângulo** é um polígono simples com exactamente três vértices convexos, chamados **cantos** (ver figura 4.1).

Para um conjunto S com n pontos no plano, uma **pseudo-triangulação** T é definida como uma partição do invólucro convexo de S em pseudo-triângulos interiores disjuntos cujos vértices são pontos de S (cada ponto de S é um vértice de T e vice-versa). Uma **pseudo-triangulação mínima** é uma pseudo-triangulação com o número mínimo de arestas de entre todas as pseudo-triangulações de S. Streinu [99] mostrou que qualquer pseudo-triangulação mínima tem $2n - 3$ arestas. Equivalentemente, podemos definir uma pseudo-triangulação mínima como uma pseudo-triangulação com um número mínimo de pseudo-triângulos.

Qualquer pseudo-triangulação mínima tem $n-2$ pseudo-triângulos e, pela fórmula

[1]Problema bastante conhecido no campo da visibilidade, que se resume ao seguinte: dada uma linha orientada e um conjunto de objectos, encontrar o primeiro objecto intersectado pela linha.

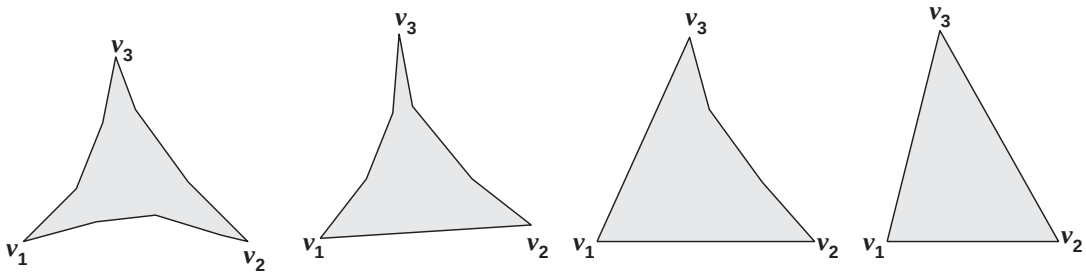


Figura 4.1: Exemplos de pseudo-triângulos. Os vértices v_1, v_2 e v_3 são cantos.

de Euler, tem-se que:

$$número\ de\ faces = número\ de\ arestas - vértices + 1 = vértices - 2. \tag{4.1}$$

Kettner et al. [53] provaram que um conjunto de pontos em posições arbitrárias, tem uma pseudo-triangulação mínima cujo máximo grau dos vértices é cinco. Randall et al. [88] determinaram expressões para o número de triangulações e de pseudo-triangulações mínimas, quando S tem apenas um ponto pertencente ao invólucro convexo. Para um conjunto de pontos em posições arbitrárias, provaram a existência de um limite superior para as pseudo-triangulações. Aichholzer et al. [2] produziram uma base de dados para todos os tipos de ordens possíveis de pseudo-triangulações para, no máximo, 10 pontos no plano.

4.1 Novos conceitos sobre pseudo-triangulações

Nesta secção introduzimos o conceito de pseudo-triangulação e mostramos alguns resultados básicos. São explorados algumas propriedades combinatórias e topológicas das pseudo-triangulações. É feita uma introdução à função característica de um polígono simples e a prova da sua propriedade aditiva com respeito às pseudo-triangulações. É apresentada uma condição necessária e suficiente para uma pseudo-triangulação ser uma triangulação.

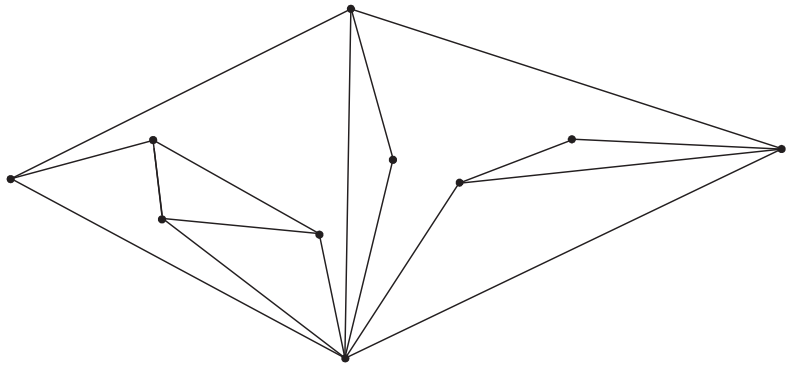


Figura 4.2: Uma pseudo-triangulação de 10 pontos.

Definição 4.1.1 *Uma* ***corda*** *de P é um segmento de recta cujos extremos são dois vértices não adjacentes de P.*

Notemos que uma diagonal de P é uma corda que tem intersecção vazia com o exterior de P, ao contrário da corda, cuja intersecção com o exterior pode não ser vazia. Dizemos que T é uma pseudo-triangulação de P se satisfaz a seguinte condição:

(i) ***Propriedade Combinatória -*** Os vértices de T são vértices de P. Cada segmento de T é uma aresta de P ou uma corda de P. Um segmento em T é incidente a exactamente um triângulo em T se for uma aresta de P, e é incidente a exactamente dois triângulos em T, se for uma corda de P. Em último caso, podemos chamar ao segmento uma corda de T. Se for uma diagonal de P, poderemos chamar-lhe diagonal de T. Além disso, T não tem pares de cordas que se intersectem.

Qualquer triangulação de P é também uma pseudo-triangulação de P. Seja T uma pseudo-triangulação de P. Fixemos uma orientação para cada triângulo T_j de T da seguinte maneira: consideramos ∂T_j como tendo a orientação na qual os três vértices são vistos na mesma ordem que em ∂P. Então T_j tem a mesma orientação que ∂T_j (ver figura 4.3). Como iremos ver no teorema 4.1.2, uma pseudo-triangulação T de P é uma triangulação de P, se e somente se satisfaz a condição (ii) abaixo:

(ii) ***Propriedade Topológica -*** Todos os triângulos em T têm a mesma orientação.

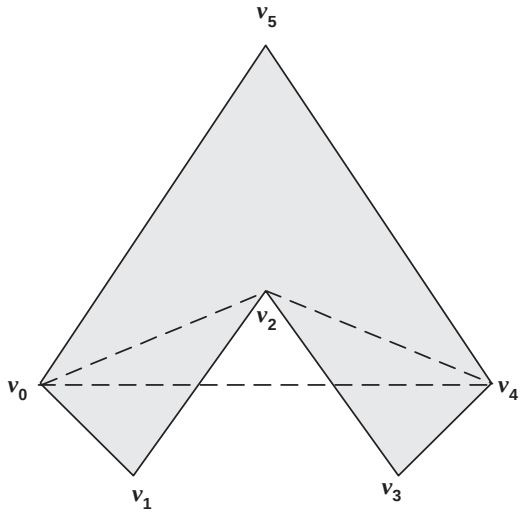


Figura 4.3: Uma pseudo-triangulação do polígono $(v_0, ..., v_5)$. Os triângulos (v_0, v_1, v_2), (v_2, v_3, v_4) e (v_0, v_4, v_5) têm orientação positiva. O triângulo (v_0, v_2, v_4) tem orientação negativa.

Existe uma outra maneira de definir pseudo-triangulação, que nos ajuda a perceber de forma intuitiva este conceito.

Seja C um polígono convexo com n vértices, onde ∂C contém a lista de vértices $u_0, u_1, u_2, ..., u_{n-1}$ numa orientação positiva (anti-horária). Então as triangulações de C estão numa correspondência unívoca com as pseudo-triangulações de P na seguinte maneira: uma triangulação T' de C corresponde a uma pseudo-triangulação T de P se (u_i, u_j) é uma diagonal de T' se e somente se (v_i, v_j) for uma corda de T. Podemos chamar a esta correspondência de mapa natural (ver figura 4.4).

Definamos uma operação chamada **troca** que transforma uma pseudo-triangulação de P noutra pseudo-triangulação. Seja T uma pseudo-triangulação de P e (v_i, v_j) uma corda de T incidente a dois triângulos, T_1 e T_2, de T. Sejam v_k e v_l outros dois vértices de T_1 e T_2. Dizemos que a corda (v_k, v_l) de P é o **dual** de (v_i, v_j) com respeito a T. Uma *troca* é a operação de reposição de uma corda pelo seu dual numa pseudo-triangulação. Esta operação é reversível. Se a operação *troca* for aplicada a polígonos convexos, ela tranformará uma triangulação do polígono noutra. Defimos também **grafo-troca** de P como sendo o grafo onde os seus nós correspondem à pseudo-triangulação de P. Dois

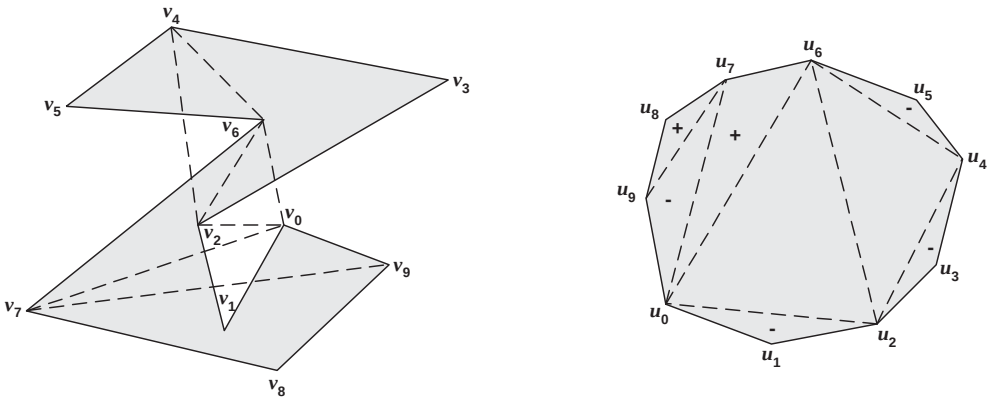


Figura 4.4: Uma pseudo-triangulação e o seu mapa natural. O sinal no triângulo (u_i, u_j, u_k) indica a orientação do triângulo (v_i, v_j, v_k).

nós do grafo são adjacentes se a correspondente pseudo-triangulação pode ser obtida de uma outra por apenas uma operação *troca*.

Seja P um polígono simples e U o conjunto dos vértices no plano $\mathbb{R}^2$. Definimos a função característica $\chi_P : \mathbb{R}^2 \setminus U \longrightarrow \{0, \pm\frac{1}{2}, \pm 1\}$ de P da seguinte maneira: para um ponto $p \in \mathbb{R}^2 \setminus U$ definimos a grandeza de $\chi_P(p)$ como sendo igual a 0 se p está no exterior de P, $\frac{1}{2}$ se p está em $\partial P \setminus U$ e 1 se p está no interior de P. O sinal de $\chi_P(p)$ é definido como sendo positivo ou negativo se a orientação de P é, respectivamente, positiva ou negativa. $\chi_P(p)$ será indefinido se p é um vértice de P.

Teorema 4.1.1 *Seja T uma pseudo-triangulação de P que contém os triângulos T_j, para $1 \leq j \leq n-2$. Denotemos por V o conjunto de vértices de P. Então para cada ponto $q \in \mathbb{R}^2 \setminus V$ temos a seguinte identidade:*

$$\chi_P(q) = \sum_{j=1}^{n-2} \chi_{T_j(q)}$$

Prova: A identidade prova-se mostrando que a operação *troca* deixa o membro da direita da igualdade invariante. Suponhamos, sem perda de generalidade, que os triângulos T_1 e T_2 de T são substituídos por dois novos triângulos T_1' e T_2' por uma

operação *troca*. Uma análise do caso, mostra facilmente que $\chi_{T_1}(q)+\chi_{T_2}(q) = \chi_{T_1'}(q)+\chi_{T_2'}(q)$ (ver figura 4.5).

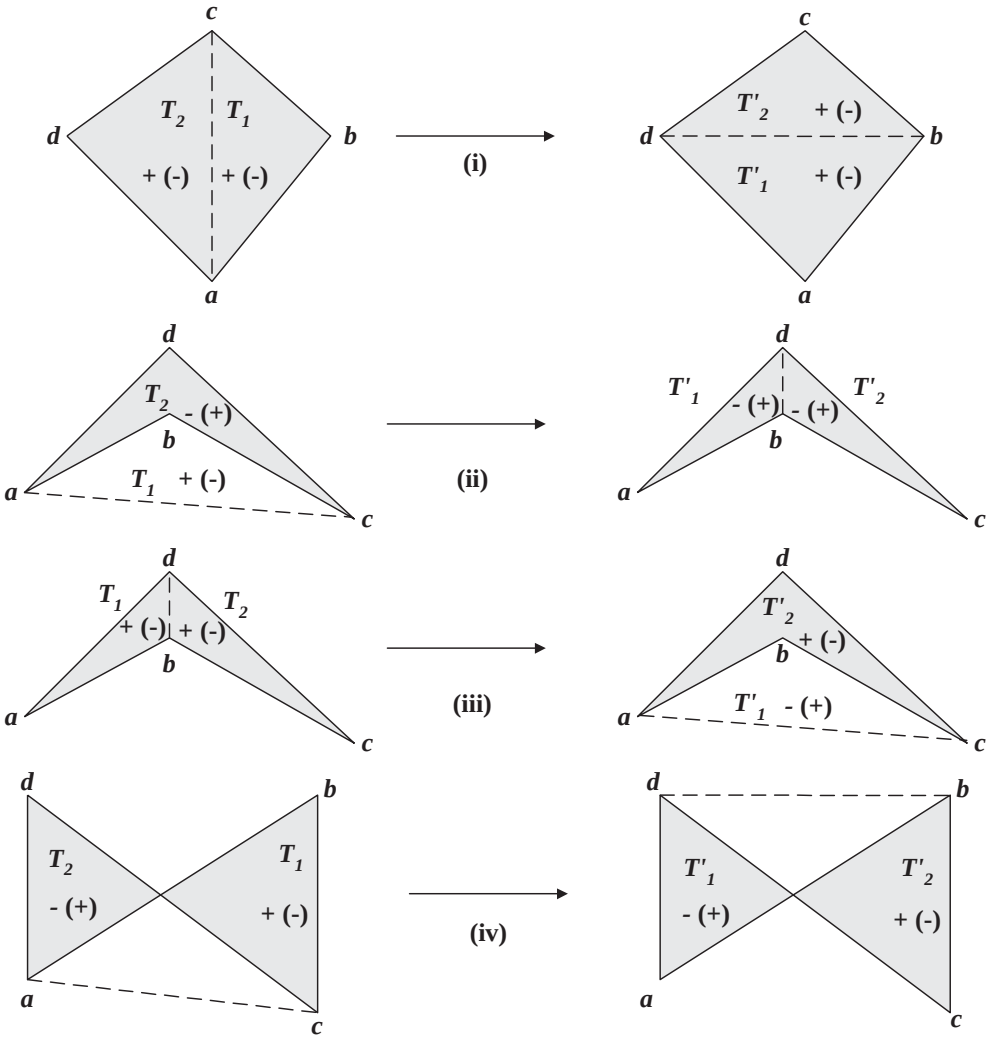


Figura 4.5: A operação *troca*. Caso (i): não há sobreposição; casos (ii) e (iii): há sobreposição total; caso (iv): há sobreposição parcial.

A prova fica completa pelos dois factos que o grafo *troca* de P é ligado e que a equação verifica-se se T for uma triangulação de P. ■

Seja P um polígono simples. Denotemos por $A(P)$, a área de P, cujo valor é a área de P e cujo sinal é dado pela orientação de P. Existe uma identidade, idêntica à do teorema 4.1.1 que é a seguinte:

$$A(P) = \sum_{j=1}^{n-2} A(T_j)$$

Uma prova similar à do teorema 4.1.1 pode ser usada para mostrar a identidade da área. (Para uma prova alternativa de um caso especial ver [69].)

Lema 4.1.1 *Em qualquer pseudo-triangulação T de P, nenhum par de diagonais se intersecta.*

Prova: Caso se intersectassem, usando o seu mapa natural, teríamos uma triangulação do polígono convexo com um par de diagonais que se intersectavam, o que é uma contradição. ■

Lema 4.1.2 (Lema da Clausura) *Suponhamos que R e Q são dois polígonos simples tal que Q não intersecta o exterior de R ($Q \subseteq R$), tem pelo menos três vértices em comum com R e os vértices restantes estão no interior de R. Então os vértices comuns de R e Q aparecem pela mesma ordem à volta de ∂R e ∂Q se R e Q tiverem a mesma orientação e aparecem por ordem inversa se R e Q tiverem orientações opostas (ver figura 4.6) .*

Prova: Usaremos a indução matemática no número de vértices não comuns a R e Q. Suponhamos que os vértices comuns a R e Q são $w_0, w_1, ..., w_{k-1}$ ordenados em ∂Q. Seja $w_k := w_0$. Primeiro assumimos que existe um vértice de R ou de Q que não é comum. Isto implica que existe um índice i, $0 \leq i < k$ tal que a porção de ∂Q desde w_i até w_{i+1} não é um subconjunto de ∂R. Seja Π essa porção de ∂Q desde w_i até w_{i+1}, que é uma cadeia poligonal orientada. Π particiona o interior de R em dois polígonos simples R_i e L_i, então R_i encontra-se à direita de Π e L_i à esquerda de Π. Assumimos que R_i e L_i têm a mesma orientação que P. Pelo Teorema da Curva de Jordan (ver capítulo 2), o interior de Q deve pertencer inteiramente a R_i ou a L_i. Se Q estiver em R_i, então por indução e pelo facto de w_i e w_{i+1} aparecerem pela mesma ordem que em ∂R_i e ∂Q, Q tem a mesma orientação que R_i e os seus vértices comuns

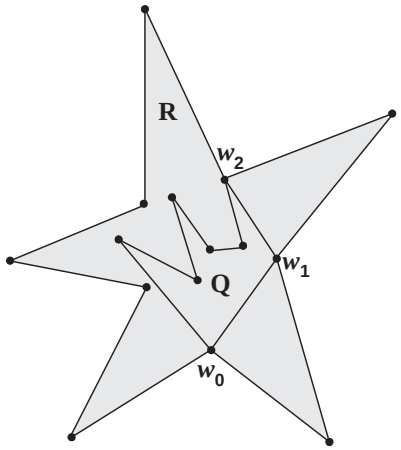


Figura 4.6: Exemplo do Lema da Clausura.

aparecem pela mesma ordem em ∂R_i e ∂Q. Como neste caso, cada vértice comum a R e Q é também comum a R_i e Q, segue-se o passo indutivo: se Q está em L_i, então, outra vez por indução, e pelo facto de w_i e w_{i+1} aparecerem por ordem contrária em ∂L_i e ∂Q, Q tem uma orientação contrária a L_i e os seus vértices comuns aparecem por ordem contrária em ∂L_i e ∂Q. Como neste caso, cada vértice comum a R e a Q é também comum a L_i e Q, segue-se o passo indutivo. O passo básico da indução, é o caso em que R e Q têm o mesmo conjunto de vértices. Neste caso, um argumento semelhante verifica-se quer para R_i ou L_i idêntico a R, para todo $0 \leq i < k$. Por outras palavras, neste caso, R e Q são o mesmo polígono tendo ambos a mesma ou orientações contrárias. ■

Corolário 4.1.1 *Suponhamos que t é um dos triângulos numa pseudo-triangulação de P. Se t tiver uma orientação negativa, então pelo menos um dos seus lados não é um diagonal nem uma aresta de P.*

Prova: Se os três lados de t forem arestas ou diagonais de P, aplica-se o lema 4.1.2 e consequentemente, t tem a mesma orientação de P, que será positiva, o que é uma contradição. ■

Teorema 4.1.2 *Seja T uma pseudo-triangulação de P. Então T é uma triangulação de P se e só se todos os triângulos de T têm uma orientação positiva.*

Prova: Se T é um triângulo orientado negativamente, pelo corolário 4.1.1 implica que T não é uma triangulação de P. Se todos os triângulos tiverem orientação positiva, então pelo teorema 4.1.1 implica que: (i) triângulos de T não podem intersectar o exterior de P; (ii) dois triângulos de T não podem ter pontos interiores em comum; (iii) cada ponto pertencente ao interior de P tanto está no interior de um triângulo de T ou na corda comum desses dois triângulos. Por outras palavras, os triângulos de T particionam o interior de P e portanto T é uma triangulação de P. ■

Uma implicação do teorema 4.1.2 é que em qualquer pseudo-triangulação T de P pelo menos um triângulo é orientado positivamente. Isto acontece, pois se todos os triângulos de T forem orientados negativamente, então T é uma triangulação do polígono idêntica a P mas com orientação contrária. Ora, isto é um absurdo.

O argumento que serve de base à implicação é que para cada pseudo-triangulação de P há uma sequência de $O(n)$ operações *troca*, que convertem a pseudo-triangulação em triangulações de P. Além disso, após cada passo, podemos determinar em tempo $O(1)$ se a pseudo-triangulação corrente é, de facto, uma triangulação de P bastando simplesmente contar quantos triângulos estão negativamente orientados na triangulação.

4.2 Pseudo-triangulações mínimas restritas

Muitas dos trabalhos de investigação em pseudo-triangulações centram-se nas propriedades e algoritmos para pseudo-triangulações mínimas para um dado conjunto de pontos ou para um dado conjunto de objectos convexos. Nestes casos, as arestas da pseudo-triangulação são escolhidas de um conjunto completo de arestas de um conjunto de pontos inicial.

É natural considerarmos algumas restrições na escolha das arestas. Nesta secção, analisamos propriedades de pseudo-triangulações minimais restritas como sendo uma subconjunto de uma dada triangulação, pseudo-triangulações mínimas restritas como sendo um super-conjunto de um dado conjunto de segmentos de recta que não se intersectam e algoritmos para encontrar essas pseudo-triangulações.

Para encontrar uma pseudo-triangulação minimal restrita numa dada triangulação, é preciso identificar as arestas que se pretendem remover. É mostrada na subsecção 4.2.2 uma propriedade para estas arestas (teorema 4.2.2). Esta propriedade permite

construir um algoritmo, apresentado na subsecção 4.2.4, de ordem linear, para encontrar uma pseudo-triangulação minimal.

Em contraste com a pseudo-triangulação de um conjunto S constituído por n pontos, onde todas as pseudo-triangulações mínimas de S têm a mesma cardinalidade, $2n-5$, o tamanho das pseudo-triangulações mínimas restritas, numa dada triangulação, T, dependem não só de n, mas também de T.

Na subsecção 4.2.3 abordam-se os possíveis tamanhos de pseudo-triangulações minimais e mínimas. Mostra-se que a razão entre os tamanhos da melhor e da pior pseudo-triangulação restrita em alguma T e o tamanho da pseudo-triangulação mínima de uma triangulação S, pode variar entre 1 e $\frac{2}{3}$. O limite inferior é assimptoticamente óptimo. Além disso, o tamanho de uma pseudo-triangulação minimal restrita numa triangulação depende da sequência da construção dos pseudo-triângulos. (Numa pseudo-triangulação minimal, cada pseudo-triângulo, foi expandido até aos seus limites; qualquer outra expansão viola a definição de pseudo-triângulo. Uma pseudo-triangulação minimal pode não ser mínima em relação a todos os possíveis pseudo-triângulos restritos nessa triangulação.) Mostra-se, também, que a razão entre o tamanho da pseudo-triangulação minimal mais pequena e o tamanho da pseudo-triangulação minimal maior restrita numa mesma triangulação pode variar entre 1 e $\frac{2}{3}$. É sabido que o tamanho de uma pseudo-triangulação mínima restrita em qualquer conjunto S e T tem pelo menos $2n-3$, [99]. Mostra-se que o número máximo de arestas nessas pseudo-triangulações é limitado por $3n-8$.

Na seccção 4.2.5, estudam-se as pseudo-triangulações que contêm um dado conjunto L de segmentos de recta que não se intersectam. Um resultado interessante é que o tamanho de uma pseudo-triangulação mínima para L depende somente do número reflexo de vértices de L. A prova usa um algoritmo para a construção dessa pseudo-triangulação mínima.

4.2.1 Preliminares

Como foi referido na subsecção 3.1.1 podemos definir uma triangulação T, de um conjunto de pontos S, do plano, como sendo o grafo máximo do plano tendo S como vértices.

A partir de aqui assume-se que os pontos estão dispostos no plano de forma

arbitrária, ou seja, não há três pontos de S colineares e que todos os ângulos são diferentes de π.

Seja T' um subgrafo de T. Para cada vértice $p \in S$ defina-se como $\alpha(p)$ o maior ângulo em p entre duas arestas vizinhas incidentes em p. Um vértice p em T' é chamado de **ponto reflexo** se $\alpha(p) \geq \pi$ em T'.

Uma pseudo-triangulação mínima de um conjunto de pontos é aquela que tiver o menor número de arestas. É sabido que o número de arestas em qualquer pseudo-triangulação mínima de n pontos é $2n - 3$, ver [99].

Seja p um pseudo-triângulo e $T(p)$ uma triangulação de p. Seja $T(p) \setminus \{p\}$ o que resta de $T(p)$ após a remoção das arestas de p. O grafo dual de $T(p)$ é definido como usualmente: cada nó no grafo corresponde à face de um triângulo em $T(p)$ e dois nós definem uma aresta do grafo se o triângulo correspondente partilham uma aresta. Uma **cadeia em estrela** consiste em três cadeias simples que partilham um mesmo nó final.

Lema 4.2.1 *O dual de qualquer triangulação de um pseudo-triângulo é uma cadeia simples ou uma cadeia em estrela.*

Prova: Vejamos a figura 4.7. Cada aresta interior da triangulação de um pseudo-triângulo atravessa duas cadeias diferentes pela não-convexidade das suas três cadeias. Isto implica que essas arestas interiores formam pelo menos um triângulo. ■

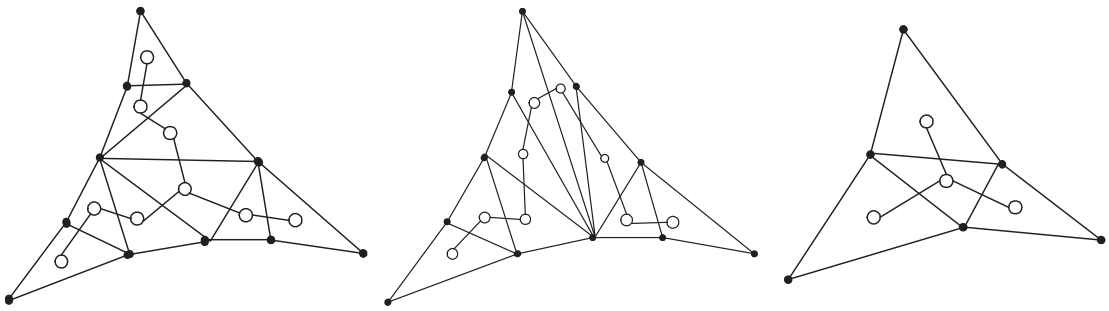


Figura 4.7: Diferentes formas do grafo dual do lema 4.2.1.

Lema 4.2.2 *Seja $T(p)$ a triangulação de um pseudo-triângulo p. Há uma correspondência perfeita entre as arestas em $T(p) \setminus \{p\}$ e os vértices reflexos de p, que fazem uma correspondência entre cada aresta e um dos seus vértices.*

Prova: Pelo lema 4.2.1, as arestas de $T(p) \setminus \{p\}$ formam uma árvore que contém exactamente um canto de p com um ciclo único, que é formado por um triângulo (ver figura 4.8). No primeiro caso, escolhemos o canto como uma raiz e direccionamos todas as arestas de $T(p) \setminus \{p\}$ na direcção contrária à da raiz. Então cada vértice reflexo terá uma aresta da árvore a apontar para ele, estabelecendo assim a desejada correspondência unívoca (correspondência entre as arestas e os vértices reflexos). Se $T(p) \setminus \{p\}$ contém um triângulo, orientamos as arestas do triângulo ciclicamente numa direcção qualquer e orientamos todas as outras arestas afastadas do ciclo. Mais uma vez, os vértices reflexos têm uma aresta da árvore a apontar para eles. ■

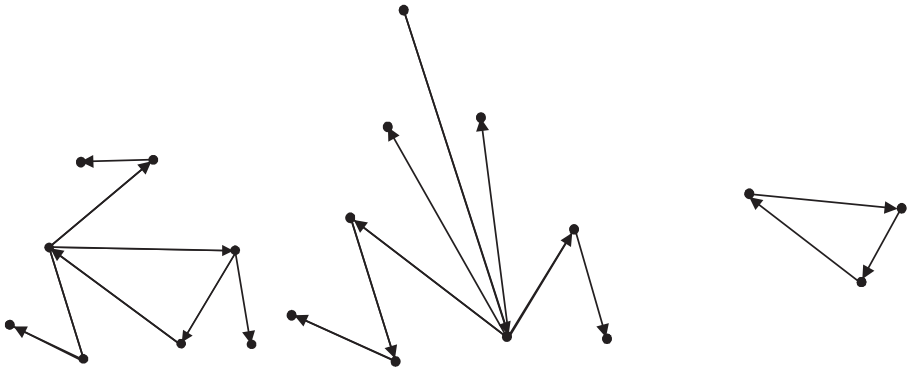


Figura 4.8: As arestas de $T(p) \setminus \{p\}$ do lema 4.2.2.

O lema 4.2.2 pode ser extendido a uma pseudo-triangulação e não se restringir apenas a pseudo-triângulos.

Teorema 4.2.1 *Seja T uma pseudo-triangulação de um conjunto de pontos, S, e seja $P \subseteq T$ uma pseudo-triangulação de S. Então existe uma correspondência perfeita entre as arestas em $T \setminus P$ e os vértices reflexos de P, que fazem uma correspondência entre cada aresta a um dos seus dois vértices.*

Prova: Qualquer vértice reflexo de P pertence exactamente a um pseudo-triângulo no qual é um vértice reflexo. Assim, podemos simplesmente aplicar separadamente o lema 4.2.2 a cada pseudo-triângulo de P. ■

O próximo lema é importante para a caracterização das pseudo-triangulações mínimas no teorema 4.2.2.

Lema 4.2.3 *Seja p um pseudo-triângulo e seja E um conjunto não vazio de arestas dentro de p que particionam p em pseudo-triângulos mais pequenos. Então verifica-se um dos seguintes casos:*

1. *E é um triângulo.*
2. *E tem uma aresta e tal que $E \setminus \{e\}$ continua a particionar p em pseudo-triângulos mais pequenos.*

Prova: Cada aresta de E liga duas cadeias reflexas diferentes de p. Se $|E| \geq 4$, então E contém pelo menos duas arestas que ligam o mesmo par de lados das cadeias reflexas de p. Escolhamos entre todas estas arestas a aresta e que está a incidir com o pseudo-triângulo que contém o canto comum destas cadeias. Se removermos e juntaremos dois pseudo-triângulos numa nova face que está limitada por porções de duas cadeias reflexas e por uma aresta entre estas cadeias. Assim, esta face é um pseudo-triângulo, e e é a aresta pretendida para o caso 2 do lema. Fica somente por provar o caso em que E tem no máximo três arestas. Este caso pode ser tratado por uma simples análise.

■

4.2.2 Pseudo-triangulações minimais

Seja T uma triangulação de S e P^T uma pseudo-triangulação restrita em T, isto é $P^T \subseteq T$. Uma pseudo-triangulação P^T é minimal, P^T_{mal}, se nenhum subconjunto próprio de P^T é uma pseudo-triangulação. P^T é uma pseudo-triangulação mínima, P^T_{min}, se contiver o menor número de arestas de todas as possíveis pseudo-triangulações restritas a T. Por simplicidade, usaremos, pseudo-triangulações restritas P^T, como uma pseudo-triangulação restrita, numa dada triangulação T.

A definição de pseudo-triangulação minimal envolve uma afirmação acerca de todos os subconjutos de arestas. O teorema seguinte, mostra que é suficiente verificar apenas um número linear de subconjuntos próprios para estabelecer que uma pseudo-triangulação é minimal.

Teorema 4.2.2 (Caracterização de pseudo-triangulações minimais) *Uma pseudo-triangulação P é minimal se e só se:*

- *não houver nenhuma aresta* $e \in P$ *tal que* $P \setminus \{e\}$ *é uma pseudo-triangulação, e*
- *se não houver uma face triangular* $\{e_1, e_2, e_3\} \in P$ *tal que* $P \setminus \{e_1, e_2, e_3\}$ *é uma pseudo-triangulação.*

Prova: A condição necessária é imeditada. Suponhamos então que $P' \subset P$ é uma pseudo-triangulação que é um subconjunto próprio de P. Pretendemos mostrar que alguma aresta do triângulo de P pode ser removida. Seja p a face do pseudo-triângulo de P' que contém alguma aresta E de $P \setminus P'$. Estas arestas subdividem p em pseudo-triângulos; podemos aplicar o lema 4.2.3 a p. Obteremos uma aresta cuja remoção levará a uma pseudo-triangulação, ou E é um triângulo, cuja remoção une quatro faces de P em p. ■

4.2.3 Razão entre os tamanhos de pseudo-triangulações

Nesta subsecção são mostradas algumas relações entre os tamanhos de T, P^T (pseudo-triangulações restritas), P^T_{mal} (minimal P^T em T), P^T_{min} (mínimo P^T em T) e $P_{min}(S)$ (pseudo-triangulação mínima de um conjunto de pontos, S).

Teorema 4.2.3 *Seja* S *um conjunto de* n *pontos em posições arbitrárias e* T *uma triangulação de* S*. O número de arestas em* P^T_{min} *é no máximo* $3n-8$*, para* $n \geq 5$*. Há um número infinito de valores de* n *para os quais a triangulação existe onde* P^T_{min} *tem* $3n-12$ *arestas.*

Prova: Suponhamos que k vértices estão no invólucro convexo de S. Então, pela equação 4.1 (equação de Euler), qualquer triangulação de T tem no máximo $3n-k-3$ arestas. Assim, quando $k \geq 5$, o limite inferior verifica-se. Facilmente se verifica que quando $n \geq 5$ e k é 3 ou 4, podemos sempre remover pelo menos $5-k$ arestas e podemos ainda obter uma pseudo-triangulação.

Uma família de triangulações que mostram o limite inferior é apresentado na figura 4.9.

O número de vértices é um múltiplo de 3 e $k = 6$. Os exemplos são construídos indutivamente, pela remoção do triângulo central e subdividindo os pseudo-triângulos

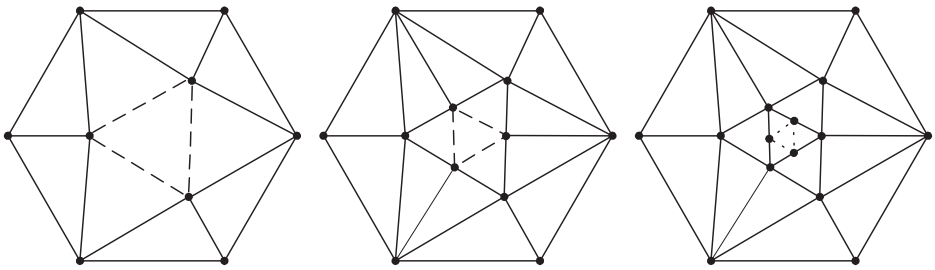


Figura 4.9: Três passos da construção indutiva do teorema 4.2.3. As três arestas do triângulo central (a tracejado) pode ser removido.

resultantes como mostra a figura 4.9. Os novos pontos serão ligeiramente ajustados no centro por forma a obter um conjunto de pontos numa posição arbitrária, e para assegurar que os caminhos directos que vão desde o centro aos vértices do hexágono exterior fazem voltas em *zigzag*. O único cojunto de arestas que pode ser removido é o triângulo central. O pseudo-triângulo resultante tem $3n - 12$ arestas. Pode-se verificar por uma análise, usando o teorema 4.2.2, que é uma pseudo-triangulação minimal. Como só havia uma maneira para obter uma pseudo-triangulação como subgrafo de T, então a pseudo-triangulação minimal é única. Portanto, é também uma pseudo-triangulação mínima. ■

Teorema 4.2.4

(a) *Há casos em S e T tal que o tamanho de T, P^T_{min} e todas as outras pseudo-triangulações P^T são iguais.*

(b) *A razão entre os tamanhos de duas diferentes pseudo-triangulações minimais restritas numa dada triangulação variam entre $\frac{2}{3}$ e $\frac{3}{2}$. Estes limites são limites assimptoticamente fechados.*

(c) *A razão entre o tamanho da pseudo-triangulação mínima de S e da pseudo-triangulação mínima restrita em T varia entre 1 e $\frac{3}{2}$, que são assimptoticamente fechados. Estes limites verificam-se para o tamanho da pseudo-triangulação minimal restrita em T.*

Prova: Os limites de variação nas razões, advêm do facto que uma pseudo-triangulação de n pontos tem entre $2n-3$ e $3n-6$. Omite-se a prova detalhada que os limites são fechados, mas mostram-se exemplos típicos na figura 4.10.

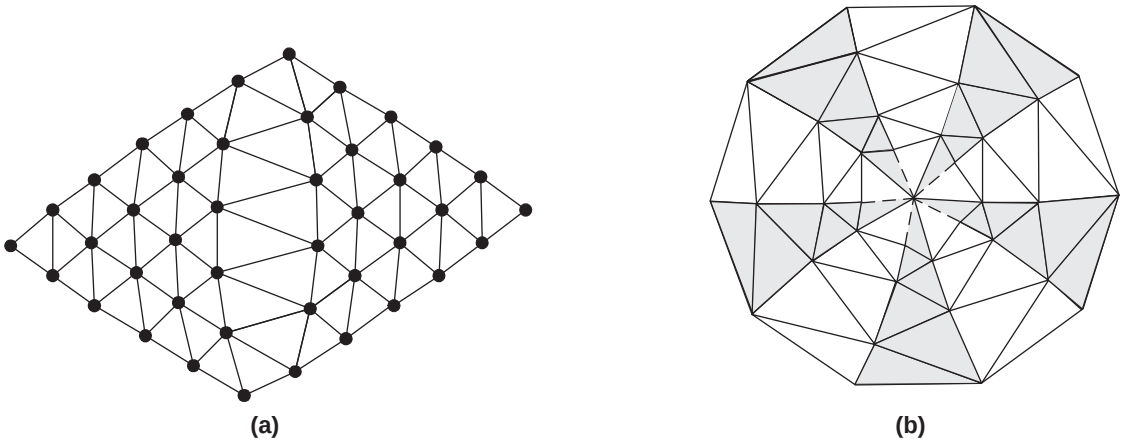


Figura 4.10: Exemplos para a prova do teorema 4.2.4.

(a) A triangulação T na figura 4.10 (a) é obtida alterando a grelha triangular por forma a que os lados fiquem em fole. Pode-se verificar por uma análise, usando o teorema 4.2.2, que é uma pseudo-triangulação minimal e, também uma pseudo-triangulação mínima em T.

(b) Na triangulação da figura 4.10 (b) podemos obter uma triangulação minimal com $3n-18$ arestas removendo as cinco arestas a tracejado no centro, ou podemos obter outra triangulação minimal com $2n-2$ arestas, removendo das zonas a cinzento.

(c) O exemplo da figura do teorema 4.2.3 é uma pseudo-triangulação mínima e minimal, $P^T(S)$ com $3n-12$ arestas. Uma pseudo-triangulação mínima de S tem sempre $2n-3$ arestas. ■

4.2.4 Construção de uma pseudo-triangulação minimal numa triangulação

Nesta subsecção, apresenta-se um algoritmo de complexidade linear para construir uma pseudo-triangulação minimal numa dada triangulação T. Pelo teorema 4.2.2, precisamos apenas de verificar onde podemos remover uma aresta ou um triângulo e manter uma pseudo-triangulação. Se for este o caso, removemos a aresta ou o triângulo e continuamos com a pseudo-triangulação resultante. O próximo lema explica como podemos executar este teste com eficiência.

Lema 4.2.4

(a) *Seja P uma pseudo-triangulação e $e \in P$ uma aresta. Então $P \setminus \{e\}$ é uma pseudo-triangulação se e só se a remoção de e forma um novo vértice reflexo, ou seja, se um extremo final de e não é reflexo em P e for reflexo em $P \setminus \{e\}$.*

(b) *Seja P uma pseudo-triangulação e $\{e_1, e_2, e_3\} \in P$ uma face triangular em P. Então $P \setminus \{e_1, e_2, e_3\}$ é uma pseudo-triangulação se e só se a remoção do triângulo tornar todos os três vértices reflexos, ou seja, se os três vértices de $\{e_1, e_2, e_3\}$ não forem reflexos em P e forem reflexos em $P \setminus \{e_1, e_2, e_3\}$.*

Prova: Remover uma aresta ou um triângulo cria uma nova face unindo dois ou quatro pseudo-triângulos, respectivamente. Temos que verificar se esta nova face contém três vértices convexos. A prova é quase imediata, pois basta contar os ângulos convexos que incidem nos vértices afectados, antes e após a remoção da aresta ou do triângulo. (No caso (a), também significa que somente um ponto extremo de e pode ser um novo vértice reflexo em $P \setminus \{e\}$). ■

Computacionalmente, as condições do lema 4.2.4 podem ser facilmente verificadas. Por exemplo, seja $e = ab$ uma aresta numa pseudo-triangulação P. Sejam α_1 e α_2 os dois ângulos incidentes a e em a, e sejam β_1 e β_2 os dois ângulos correspondentes em b. Então $P \setminus \{e\}$ é uma pseudo-triangulação se e só se $\alpha_1 < \pi$, $\alpha_2 < \pi$ e $\alpha_1 + \alpha_2 > \pi$, ou se $\beta_1 < \pi$, $\beta_2 < \pi$ e $\beta_1 + \beta_2 > \pi$. A condição pode ser formulada de forma similar, para a remoção de um triângulo (lema 4.2.4 (b)). Assim, o algoritmo pode ser executado num tempo constante, para uma dada aresta ou triângulo, que possa ser removida/o.

O algoritmo para a construção de uma pseudo-triangulação minimal funciona, então, da seguinte maneira: chamamos uma aresta ou um triângulo removíveis se for satisfeita a condição (a) ou (b) do lema 4.2.4, respectivamente. Começamos com a triangulação dada. O algoritmo mantém uma lista de todas as arestas removíveis, que é inicializada num tempo linear por pesquisa de todas as arestas. Quando existir uma aresta removível, simplesmente removemos esta aresta e actualizamos a lista das arestas removíveis. A remoção de uma aresta $e = [ab]$ pode afectar o estatuto de removíveis de no máximo quatro arestas da pseudo-triangulação corrente P (nomeadamente, duas arestas vizinhas em a e em b). Estas arestas podem ser verificadas num tempo constante.

Repete-se este procedimento até a lista de arestas removíveis ficar vazia. Agora verificamos se há algum triângulo removível de acordo com a condição do lema 4.2.4 (b), e removêmo-lo. Podemos mostrar facilmente que a remoção de um triângulo não pode criar uma nova aresta removível ou um novo triângulo removível. Assim, podemos simplesmente percorrer todas as faces de P sequentemente, num tempo linear.

No fim, obtemos uma pseudo-triangulação sem termos removido arestas ou triângulos, que é uma pseudo-triangulação minimal, pelo teorema 4.2.2. Temos, assim, o seguinte teorema:

Teorema 4.2.5 *O algoritmo produz uma pseudo-triangulação minimal P^T_{mal} de uma triangulação T, dada, num tempo linear.*

4.2.5 Construção de uma pseudo-triangulação contendo um dado conjunto de arestas

Nesta subsecção, encontramos uma pseudo-triangulação mínima que contém um conjunto L dado, de segmentos que não se intersectam. A ideia básica é manter o conjunto de vértices reflexos do grafo $G(S, L)$ dado invariante, quando adicionamos arestas extras a L para a construção da pseudo-triangulação de L [84].

Teorema 4.2.6 *Para qualquer conjunto L de segmentos que não se intersectam, existe uma pseudo-triangulação $T'_L(S) \supseteq L$ que tem o mesmo conjunto de vértices reflexos que $G(L, S)$.*

Prova: Provamos este facto, adicionando gradualmente arestas ao conjunto L até termos a pseudo-triangulação. Primeiro adicionamos todas as arestas do invólucro convexo a L. Isto não altera o conjunto de vértices reflexos.

Depois, o conjunto L de arestas particionam o interior do invólucro convexo em faces, que podem ser consideradas independentes. Consideremos, então, uma face singular F (ver figura 4.11).

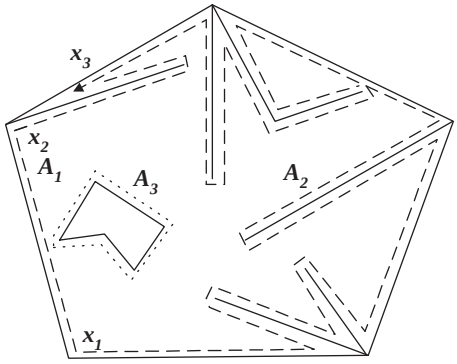


Figura 4.11: Ilustração da prova do teorema 4.2.6.

A fronteira de F tem uma componente B que é o exterior da fronteira de F, e terá, possivelmente, várias outras componentes dentro de F. Notemos que B é um ciclo único de arestas quando percorremos a fronteira de F dentro de B, embora este ciclo possa visitar a mesma aresta duas vezes (por dois lados diferentes) ou pode visitar um vértice várias vezes. Apesar de tudo, tratamos B como se fosse um polígono simples. Repetindo os seguintes passos, iremos dividir F em pseudo-triângulos:

- escolha um canto x_1 em B e percorra no sentido horário, (ao longo de B) até encontrar os dois próximos cantos x_2 e x_3 em B (B deve conter pelo menos três cantos). Denotamos o caminho desde x_1 via x_2 até x_3 ao longo de B por A_1, e a restante parte de B por A_2. Por A_3, denotamos o conjunto (possivelmente vazio) das componentes interiores da fronteira de F (ver figura 4.11).

- Encontre o caminho mais curto, S, desde x_1 até x_3 em F que é igual ao caminho A_1 desde x_1 até x_3. Por outras palavras, tomamos o caminho mais curto desde x_1 até x_3 em F, que separará A_1 de $A_2 \cup A_3$.

Este caminho S é constituído pelas seguintes componentes:

(a) uma componente inicial que acompanha alguma parte de B desde x_1 em direcção a x_2, virando à esquerda;

(b) um segmento de recta ligado que passa pelo interior de F;

(c) uma parte da fronteira do invólucro convexo de $A_2 \cup A_3$, virando à direita;

(d) um segmento de recta ligado que passa pelo interior de F; e

(e) uma componente final que acompanha alguma parte de B desde x_2 até x_3, virando à esquerda.

Qualquer uma das componentes (a), (c) ou (e) pode faltar. Se for a componente (c), então existe apenas um segmento ligado em vez de (b) e (d), advindo daí que a região que é eliminada por este caminho (à esquerda de S) é um pseudo-triângulo que não contém pontos no seu interior. Pode acontecer que S tenha apenas uma cadeia reflexa desde x_1 até x_3 ao longo de B. Neste caso, F é um pseudo-triângulo vazio. Também nenhum vértice reflexo será destruído por alguma aresta de S. S passará pelos vértices reflexos (excepto os pontos extremos x_1 e x_3), e fará viragens à esquerda quando passa à volta de uma componente que está à esquerda, e similarmente para viragens à direita.

■

Um corolário imediato do teorema extende os resultados conhecidos para $L = \emptyset$, quando $r = n$.

Corolário 4.2.1 *Todas as pseudo-triangulações mínimas de um conjunto S com n pontos, que contém um dado conjunto L de arestas com r vértices reflexos, tem $2n-r-2$ pseudo-triângulos e $3n - r - 3$ arestas.*

Capítulo 5

Alguns problemas em aberto

Talvez pelo facto desta área da Matemática ser relativamente recente, são muitos e variados os problemas que permanecem em aberto. Devido às constantes evoluções e descobertas na área da Geometria Computacional, é difícil garantir que um ou outro problema apresentado como aberto, já não tenha sido resolvido até ao momento. No entanto, *arriscamos* a apresentar alguns que até ao momento não temos conhecimento de já terem sido solucionados.

Nesta dissertação apresentou-se uma classificação de polígonos simples, com o objectivo de encontrar características semelhantes nos diversos tipos de polígonos. Classificação essa que poderá permitir, por exemplo, o desenvolvimento de outros algoritmos de partição. É natural que esta classificação possa vir ser melhorada.

Diferentes métodos de triangulação poderão melhorar a decomposição de um polígono, como tal, são procurados ainda hoje, outras abordagens na triangulação. Como foi referido na secção 3.1, permanece ainda por desenvolver um algoritmo de triangulação de polígonos simples arbitrários, de fácil implementação, de complexidade linear. Relembremos que o algoritmo de Chazelle [16], apesar de ter complexidade linear é de difícil implementação. Um interessante problema que permanece também em aberto, é a tentativa de encontrar uma decomposição num número mínimo de polígonos estrelados como foi referido na subsecção 3.3.1. Ainda na partição de polígonos, mas por quadrangulação (tratado na secção 3.5), também muitos problemas continuam em aberto, por exemplo, serão $\lfloor n/4 \rfloor$ pontos interiores de Steiner alguma vezes necessários para quadrangular um $n - \acute{a}gono$ simples? Ainda não se conhecem limites inferiores não triviais para este problema. O número mínimo de de pontos de Steiner necessários

para quadrangular um polígono simples, sobre todas a triangulações é também um problema em aberto. Será que é possível decidir se um conjunto de pontos admite uma quadrangulação onde cada quadrilátero é convexo? Esta questão surgiu por Joe Mitchell em 1993. Um outro problema de partições clássicas que permanece em aberto, respeita à classe dos polígonos ortogonais. Qual será o número mínimo de subpolígonos convexos que são necessários para cobrir um polígono ortogonal? Já no tema da pseudo-triangulação (assunto abordado no capítulo 4), questões como encontrar uma pseudo-triangulação mínima restrita numa triangulação T, constatar se este é um problema NP-difícil (NP-*hard*, em inglês), ou aprofundar assuntos sobre pseudo-triangulações mínimas sujeitas a outras restrições, continuam por ser esclarecidas.

Outros problemas, não directamente relacionados com os temas abordados na dissertação, permanecem também por resolver:

- Existirá um algoritmo linear para determinar o caminho mais curto num polígono simples, sem primeiro aplicar um algoritmo de triangulação mais complicado?

- Dado um conjunto de n pontos no plano, poder-se-á gerar polígonos simples num tempo polinomial, tendo esses pontos como vértices do polígono? Para alguns casos é possível (por exemplo, polígonos monótonos [112]), mas para um polígono qualquer a questão permanece ainda em aberto.

- Poderá uma triangulação de peso mínimo de um conjunto de pontos do plano ser encontrada num tempo polinomial? O peso de uma triangulação é o comprimento total das arestas. Este problema é um dos poucos de Garey e Johnson [37] cuja complexidade permanece desconhecida. O resultado obtido pelos melhores algoritmos de aproximação é uma constante vezes o tamanho óptimo [66]; são conhecidas boas heurísticas [25]. Se forem permitidos pontos de Steiner, são conhecidos, também, factores constantes de aproximação [30, 20], mas permanece em aberto se existirá uma triangulação de peso mínimo usando pontos de Steiner.

- Quanto espaço é necessário para determinar o invólucro convexo de uma cadeia poligonal simples ou de um polígono simples num tempo linear? Mais precisamente, dado um conjunto n de pontos ordenados ao longo da cadeia num vector A, o algoritmo deverá rearranjar os pontos no vector e o resultado será um número h de tal forma que os primeiros h elementos do vector resultante, são os pontos ordenados do invólucro convexo. O objectivo é minimizar o espaço de

armazenamento extra necessário para além do vector A, digamos para $O(log\, n)$ ou idealmente para $O(1)$.

- Será que cada par de polígonos com área igual têm uma dissecação articulada? A dissecação de um polígono A num outro B, é a partição de A num número finito de partes que podem ser reorganizadas para formar B. Uma dissecação articulada é uma dissecação onde as partes são articuladas nos vértices e a reorganização é obtida rodando as partes em torno dos seus eixos (vértices) no plano dos polígonos.

- Para uma conjunto de pontos S no plano, será que o número de pseudo-triangulações mínimas é pelo menos o número de triangulações?

- Será verdade que dois conjuntos de n pontos do plano numa posição arbitrária, com o mesmo número de pontos nos seus invólucros convexos, têm triangulações compatíveis? Duas triangulações dizem-se compatíveis se tiverem a mesma estrutura combinatória, isto é, consideremos dois conjuntos finitos, S_1 e S_2 de pontos do plano numa posição arbitrária. Duas triangulações T_1 de S_1 e T_2 de S_2 são compatíveis se as suas faces, formadas pelos triângulos, arestas e vértices são isomorfas.

Bibliografia

[1] P. K. Agarwal, J. Bash, L. J. Guibas, J. Hershberger e L. Zhang, *Deformable free space tilings for Kinetic collision detection.*, In B. R. Donald, K. Lynch e D. Rus (eds.), Algorithmic and Computacional Robotics: New Directions (Proc. 5th Workshop Algorithmic Found. Robotics), 83-96, A. K. Peters, (2001).

[2] O. Aichholzer, F. Aurenhammer e H. Krasse, *Enumerating order types for small sets applications.*, In Proc. 16th Annu. ACM Sympos. Computacional Geometry, 11-18, (2000).

[3] O. Aichholzer, M. Hokmann, B. Speckmann, e D. Tóth, *Degree bounds for constrained pseudo-triangulations.*, In: Proc. 15th CCCG, (2003).

[4] D. J. Allman, *A quadrilateral finite element including vertex rotations for plane elasticity analysis.*, International Journal for Numerical Methods in Engineering, 26: 717-730, (1988).

[5] E. Arkin, M. Held, J. Mitchell, e S. Skiena, *Hamiltonian triangulations for fast rendering*, In J. van Leeuwen, editor, Algorithms-ESA'94, LNCS 855: 36-47, Utrecht, The Netherlands, (1994).

[6] Ta. Asano, Te. Asano e H. Imai, *Partitioning a polygonal region in trapezoids*, J. ACM, 33: 290-312, (1986).

[7] Ta. Asano, Te. Asano e R.Y. Pinter, *Polygon triangulation: Efficiency and minimality*, J. Algorithms 7, 221-231, (1986).

[8] D. Avis, G. Toussaint, *An Efficient Algorithm for Decomposing a Polygon into Star-Shaped Polygons*, Pattern Recognition, 13(6): 395-398, (1981).

[9] Mark de Berg, Marc van Kreveld, Mark Overmars e Otfried Schwarzkopf *Computacional Geometry: Algorithms and Applications*, Second Revised Edition, Springer, (2000).

[10] M. Bern e D. Eppstein, *Mesh generation and optimal triangulation*, in F. K. Hwang and D.-Z. Du, editors, Computing in Euclidean Geometry. World Scientic, (1992).

[11] M. Bern e J. R. Gilbert, *Drawing the planar dual*, In Information Processing Letters, 43: 7-13, (1992).

[12] P. Bose e G. T. Toussaint, *No quadrangulation is extremely odd,* in Proc. of the International Symposium on Algorithms and Computationl Cairns, Australia, December (1995).

[13] T. Calvert, S. Xie e B. Bhattacharya, *Planning views for the incremental construction of body models*, Seventh International Conference on Pattern Recognition, 154-157, (1986).

[14] B. Chazelle, *A theorem on polygon cutting with applications*, Proc.23rd IEEE Symposium on Foundations of Computer Science, Chicago, 339-349, (1982).

[15] B. Chazelle, *Computacional Geometry and convexity*, Ph. D. thesis, Yale University, (1980).

[16] B. Chazelle, *Triangulating a simple polygon in linear time*, Discrete and Computacional Geometry, 6: 485-524, (1991).

[17] B. Chazelle e D. P. Dobkin, *Decomposing a polygon into its convex parts*, 11th ACM Symp., Theory of Computing, 38-48, (1979).

[18] B. Chazelle e D. P. Dobkin, *Optimal convex decompositions*, Computacional Geometry, 63-133, North-Holland, Amsterdam, Netherlands, (1985).

[19] B. Chazelle e J. Incerpi, *Triangulation and shape-complexity*, ACM Trans. of Graphics 3(2): 135-152, (1984).

[20] S. -W. Cheng e K. -L. Lee, *Quadtree decomposition, Steiner triangulation, and ray shooting*, In ISAAC: 9th Internat. Sympos. algorithms Computation, 367-376, (1998).

[21] V. Chvatal, *A combinatorial theorem in plane geometry*, J. Combin. Theory Ser. B, 18: 39-41, (1975).

[22] K. Clarkson, R. E. Tarjan e C. J. Van Wyk, *A fast Las Vegas algorithm for triangulating a simple polygon*, Discrete Computacional Geometry 4: 423-432, (1989).

[23] R. Connelly, E. D. Demaine e G. Rote, *Straightening polygonal arcs and convexifying polygonal cycles*, In Proc. 41th Annu. Sympos. on Found. of Computer Science, 432-442, (2000).

[24] O. Devillers, *Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems*, Internat. J, Computacional Geometry Appl., 2: 97-111, (1992).

[25] M. T. Dickerson, S.A. McElfresh e M. H. Montague, *New algorithms and empirical findings on minimun weight triangulation heuristics*, In Proc. 11th Annu. ACM Sympos. Computacional Geometry, 238-247, (1995).

[26] David Eberly, *Triangulation by ear clipping*, www.magic-software.com, (2002).

[27] H. Edelsbrunner, J. O'Rourke, e E. Welzl, *Stationing guards in rectilinear art galleries.*, Computer Vision, Graphics and Image Processing, 27: 167-176, (1984).

[28] H. ElGindy e G. T. Toussaint, *On geodesic properties of polygons relevant to linear time triangulation*, Visual Comput. 5 (1/2), 68-74, (1989).

[29] H. ElGindy e G. T. Toussaint, *On triangulating palm polygons in linear time*, Proc. Computer Graphics International´88, 308-317, Maio (1988).

[30] D. Eppstein, *Approximating the minimum weight Steiner triangulation*, Discrete Computacional Geometry, 11: 163-191, Maio (1994).

[31] H. Everett, W. Lenhart, M. Overmars, T. Shermer, e J. Urrutia, *Strictly convex quadrilateralizations of polygons.*, In Proc. of the 4th Canadian Conference of Computacional Geometry, 77-82, St. Johns, Newfoundland, (1992)

[32] H. Y. F. Feng e T. Pavlidis, *Decomposition of polygons into simpler components: feature generation for syntatic pattern recognition*, IEEE Trans. Comput., C-24, 636-650, (1975).

[33] H. Y. Feng e T. Pavlidis, *Shape discrimination*, Syntatic Pattern Recognition, 125-145, Springer Verlag, (1977).

[34] L. Ferrari, P. V. Sankar e J. Sklansky, *Minimal rectangular partitions of digitized blobs*, Proc. 5th Int. Conf. Pattern Recognition, 1040-1043, Miami Beach, (1980).

[35] S. Fisk, *A short proof of Chvatal´s watchman theorem*, J. Combin. Theory Ser. B, 24: 374, (1978).

[36] A. Fournier e D. Y. Montuno, *Triangulation simple polygons and equivalent problems*, ACM Trans. of Graphics 3: 153-174, (1984).

[37] M.R. Garey e D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP - Completeness*, W. H. Freeman, New York, NY, (1979).

[38] M.R. Garey, D.S. Johnson, F. P. Preparata e R.E. Tarjan, *Triangulating a simple polygon*, Inform. Process. Lett. 7, 175-179, Springer Verlag, (1978).

[39] S. K. Ghosh, A. Maheshwari, S. P. Pal, S. Saluja e C. E. Veni Madhavan, *Computing the shortest path tree in a weak visibility polygon*, Found. Software Tech. Theoret. Comput. Sei. 11, 369-389, (1991).

[40] J. Goodman e J. O´Rourke, *Handbook of Discrete and Computacional Geometry*, CRC Press, 429-444,(1997).

[41] R. L. Graham, *An efficient algorithm for determining the convex hull of a finite planar set*, Info. Proc. Lett., 1: 132-133,(1972).

[42] D. H. Greene, *The decomposition of polygons in convex parts*, Advances in Computing Research (F.P. Preparata, ed.), JAI Press, 235-259, (1983).

[43] P. J. Hefferman, *Linear-time algorithms for weakly-monotone poligons*, Computacional Geometry 3, 121-137, (1993).

[44] E. Heighway, *A mesh generator for automatically subdividing irregular polygons into quadrilaterals.* IEEE Transactions on Magnetics, 19(6): 2535-2538, (1983).

[45] S. Hertel e K. Mehlhorn *Fast triangulation of simple polygon*, Proc. 4th Internat. Conf. Found. Comput. Theory, Lecture Notes in Computer Science, (158): 207-218, (1983).

[46] S. Hertel e K. Mehlhorn *Fast triangulation of of the plane with respect to simple polygons*, Inform. Control 64 (1-3): 52-76, (1985).

[47] B. Joe, *Quadrilateral mesh generation in polygonal regions.* Computer Aided Design, 27(3): 209-222, (1995).

[48] B. P. Johnston, J. M. Sullivan, e A. Kwasnik, *Automatic conversion of triangular fnite meshes to quadrilateral elements.* International Journal of Numerical Methods in Engineering, 31(1): 67-84, (1991).

[49] J. Kahn, M. Klawe e D. Kleitman. *Traditional galleries require fewer watch men.* SIAM Journal of Algorithms and Discrete Methods, 4(2): 194-206, (1983)

[50] J. Mark Keil, *Decomposing a polygon into a simpler components*, SIAM J. Comp. 14, 799-817, (1985).

[51] J. Mark Keil, *Polygon Decomposition*, (1996).

[52] J. M. Keil e J. -R. Sack, *Minimun decompositions of polygonal objects*, Computacional Geometry, 63-133, North-Holland, Amsterdam, Netherlands (1985).

[53] L. Kettner, D. Kirkpatrick e B. Speckmann, *Tight degree bounds for pseudo-triangulations of points*, In Proc. 13th Canad. Conf. Computacional Geometry, 117-120, (2001).

[54] D. G. Kirkpatrick, M. M. Klawe e R.E. Tarjan, *Polygon triangulation in O(nloglogn time with sample data structures*, Discrete Computacional Geometry 7, 329-346, (1992).

[55] D. Kirkpatrick, J. Snoeyink e B. Speckmann *Kinetic collision detection for simple polygons*, In Proc. 16th Annu. ACM Sympos. Computacional Geometry, 322-330, (2000).

[56] D. Kirkpatrick e B. Speckmann, *Kinetic maintenace of context-sensitive hierarchical representations for disjoint simple polygons*, In Proc. 18th Annu. ACM Sympos. Computacional Geometry, 179-188, (2002).

[57] G. T. Klincsek, *Minimal triangulations of polygonal domains*, Discrete Math, (9): 121-123, (1980).

[58] V. Klee and P. van den Driessche, *Linear algorithms for testing the sign stability of a matrix and for finding z-maximum matchings in acyclic graphs*, Numerische Mathematik. (28): 273-285, (1977)

[59] X. Kong, H. Everett, G.T. Toussaint *The Graham scan triangulates simple polygons*, Pattern Recognition Letters, (11): 713-716, (1991).

[60] A. A. Kooshesh, B. M. E. Moret, *Three-coloring the vertices of triangulated simple polygon*, Pattern Recognition, 25, (1992).

[61] M. A. Krasnosel´ski, *Sur un critere pour qu´un damaine soit etoile*, Math. Sb. 61, (1946).

[62] R. Kuc, M. Siegel, *Efficient representation of reflecting structures for a sonar navigation model*, Proceedings of the 1987 IEEE International Conference on Robotics and Automation, 1916-1923, (1987).

[63] J. C. Latombe, *Robot motion planning*, Kluwer Acad. Publ., Boston, MA, (1991).

[64] S. H. Lee e K. Y. Chwa *A new triangulation-linear class of simple polygons*, Internat. J. Comput. Math. 22, 135-147, (1987).

[65] D.T. Lee e F.P. Preparata, *Location of a point in a planar subdivision and its applications*, SIAM Journal on Computing 6, 594-606, (1977).

[66] C. Levcopoulos e D. Krznaric, *Quasi-greedy triangulations approximating the minimum weight triangulation*, In Proc. 7th ACM-SIAM Sympos. Discrete Algorithms, 392-401, (1996).

[67] M. Levoy, *A hybrid ray tracer for rendering polygon and volume data*, IEEE Comput. Graph. Appl., (10): 33-40, (1990).

[68] E. Lodi, F. Luccio, C. Mugnai, e L. Pagli, *On two-dimensional data organization*, I. Fundam. Inform., (2): 221-226, (1979).

[69] A.M. Lopshits, *Computation of Areas of Oriented Figures*, (1965).

[70] A. Lubiw, *Decomposing polygonal regions into convex quadrilaterals.* In Proc. of the 1st ACM Symposium on Computational Geometry, 97-106, (1985).

[71] A. Lubiw, *The boolean basis problem and how to cover some polygons by rectangles*, SIAM J. on Discrete Mathematics, (3): 98-115, (1990).

[72] K. Maruyama, *A study of visual shape perception*, Department of computer Science, University of Illinois, Urbana, (1972).

[73] K. Mehlhorn, *Data Structures and Algorithms*, Volume 3: Multi-Dimensional Searching and Computacional Geometry, Springer-Verlag, Berlin (1984).

[74] G.H. Meister, *Polygons have ears*, American Mathematical Monthy 82, 648, (1975).

[75] Joseph S. B. Mitchell e Joseph O´Rourke, *Computacional Geometry Column 42*, http://maven.smith.edu/ orourke/TOPP/.

[76] D. Moitra *Finding a minimal cover for binary images: an optimal parallel algorithm*, Algorithmica, (6): 624-657, (1991).

[77] J. O´Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, (1987).

[78] J. O´Rourke, *Computacional Geometry in C*, Cambridge University Press, Cambridge, Second Edition, (1998).

[79] S. P. Pal, *Weak visibility and related problems on simple polygons*, PhD thesis, Indian Institute of Science, India, (1990).

[80] T. Pavlidis, *A review of algorithms for shape analysis*, Comp. Graphics and Image Processing, (7): 243-258, (1978).

[81] T. Pavlidis, *Structural Pattern Recognition*, Springer-Verlag, Berlin Heidelberg, (1977).

[82] T. Lozano-Perez e M. A. Wesley, *An algorithm for planning collision-free paths among polyhedral obstacles*, Commum. ACM (22): 560-570, (1979).

[83] M. Pocchiola e G. Vegter, *Pseudo-triangulations: Theory and applications*, In Proc. 12th Annu. ACM Sympos. Computacional Geometry, 291-300, (1996).

[84] M. Pocchiola e G. Vegter, *Topologically sweeping visibility complexes via pseudo-triangulations*, Discrete Computacional Geometry, 16(4): 419-453, (1996).

[85] F. P. Preparata e M. I. Shamos, *Computacional Geometry: An Introduction*, Springer-Verlag, (1985).

[86] Suneeta Ramaswami, Pedro A. Ramos e Godfried T. Toussaint, *Converting triangulations to quadrangulations.* Proc. 7th Canad. Conf. Computational Geometry, Aug 1995, 297-302, Computational Geometry Theory and Applications, 9(4): 257-276, (1998).

[87] D. Randall, G. Rote, F. Santos e J. Snoeyink. *Counting triangulations and pseudo-triangulations of wheels*, In Proc. 13th Canad. Conf. Computacional Geometry, 117-120, (2001).

[88] D. Randall, G. Rote, F. Santos e J. Snoeyink. *Counting triangulations and pseudo-triangulations of wheels*, In Proc. 13th Canad. Conf. Computacional Geometry, 149-152, (2001).

[89] D. Randall, F. Santos e I. Streinu. *Expansive motions and the polytope of pointed-triangulations*, Manuscript, FU-Berlin, September (2001).

[90] G. Rote, C. A. Wang, L. Wang, and Y. Xu. *On constrained minimum pseudotriangulations*, In Proc. 9th COCOON, (2003).

[91] J.-R Sack, *An $O(n \log n)$ algorithm for decomposing simple rectilinear polygons into convex quadrilaterals.* In Proc. 20th Annual Allerton Conf. on Communications, Control and Computing, 64-75, Urbana, Illinois, (1982).

[92] J.-R Sack and G. T. Toussaint, *A linear time algorithm for decomposing rectilinear star-shaped polygons into convex quadrilaterals.* In Proc. 19th Annual Allerton Conf. on Communications, Control and Computing, 21-30, Urbana, Illinois, (1982).

[93] J.-R Sack and G. T. Toussaint, *Guard placement in rectilinear polygons.* In G. T. Toussaint, editor, Computational Morphology, 153-175, North-Holland, (1988).

[94] A. Schoone, J. Van Leeuwen *Triangulating a star-shaped polygon*, Tech. Report, RUV-CS-80-3, University of Utrecht (1990).

[95] R. Seidel, *A simple and fast incremental randomized algorithm for computing trapezoidal decomposition and for triangulating polygons*, Computacional Geometry 1, 51-64, (1991).

[96] B. Shachter, *Decomposition of polygons into convex sets*, IEEE Trans. Comput., C-27(11): 1078-1082, (1978).

[97] T. C. Shermer, *Recent results in art galleries*, Proc. IEEE, 80(9): 1384-1399, (Setpember 1992).

[98] V. Srinivasan, L. R. Nackman, J-M. Tang, e S. N. Meshkat. *Automatic mesh generation using the symmetric axis transformation of polygonal domains.* Proceedings of the IEEE (Special Issue on Computational Geometry), 80(9): 1485-1501, (1992).

[99] I. Streinu, *A combinatorial approach to planar non-colliding robot arm motion planning*, In Proc. 41st Annu. IEEE Sympos. Found. Comput. Sci., 443-453, (2000).

[100] I. Streinu, *Folding carpenter´s rulers, robot arms, proteins: a rigidity theoretic approach*, Invited talk, 10th Annual Fall Workshop on Computacional Geometry, (2000).

[101] R. E. Tarjan e C. J. Van Wyk, *An O(nloglogn) time algorithm for triangulating a simple polygon*, SIAM, J. Comput. 17, (1988), 143-178. Erratum in 17 (1988), 106.

[102] S. B. Tor e A. E. Middleditch, *Convex decomposition of simple polygons*, ACM Trans. Graph. (3): 244-265, (1984).

[103] G. Toussaint, *A hierarchy of simple polygons*, manuscript in preparation.

[104] G. Toussaint, *A new linear algorithm for triangulating monotone polygons*, Pattern Recognition Letters 2, 155-158, (1984).

[105] G. Toussaint, *Anthropomorphic polygons*, American Mathematical Monthy, 98, 31-35, (1991).

[106] G. Toussaint, *Computing geodesic properties inside a simple polygon*, Revue D´Intelligence Artificielle, (3): 9-42, (1989).

[107] G. Toussaint, *Efficient triangulation of simple polygons*, The Visual Computer, 7(5-5): 280-295 (1991).

[108] G. Toussaint, *Quadrangulations of planar sets.*, In Proceedings of the 4th International Workshop on Algorithms and Data Structures, Springer-Verlag, 218-227, (1995).

[109] G. Toussaint, *Pattern recognition and geometrical complexity*, Proc. Fifth Inter. Conf. on Pattern Recognition, 1324-1347, (1980).

[110] G. Toussaint e D. Avis, *On a convex hull algorithm for polygons and its application to triangulation problems*, Pattern Recognition 15, 23-29, (1982).

[111] T.C. Woo e S. Y. Shin, *A linear time algorithm for triangulating a point-visible polygon*, ACM Trans. of Graphics 4(1): 60-69, (1985).

[112] C. Zhu, G. Sundaram, J. Snoeyink e J.S.B. Mitchell, *Generating random polygons with given vertices*, Computacional Geometry. Theory Appl., 6: 277-290, (1996).

Índice